Advanced Transport Protocols for Next Generation Heterogeneous Wireless Network Architectures

A Thesis
Presented to
The Academic Faculty

by

Özgür Barış Akan

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in Electrical and Computer Engineering

School of Electrical and Computer Engineering
Georgia Institute of Technology
April 2004

Advanced Transport Protocols for Next Generation Heterogeneous Wireless Network Architectures

Approved by:

Professor Ian F. Akyildiz,
Committee Chair

Professor Paul G. Steffes

Professor Chuanyi Ji

Professor Robert Braun

Professor Raghupathy Sivakumar

Date Approved: 2 April 2004

To my parents,

Fazile Akan and Abdulcelil Akan.

ACKNOWLEDGEMENTS

I would like to express my gratitude and sincere thanks to my advisor, Dr. Ian F. Akyildiz. I am indebted to Dr. Akyildiz for his invaluable and seamless support, guidance, friendship, and trust throughout my doctoral study. Dr. Akyildiz has taught me to never forget that success comes with Hard work, Integrity, Persistence, and Talent (HIP-T).

My cordial thanks also extend to Dr. Raghupathy Sivakumar, Dr. Chuanyi Ji, Dr. Paul G. Steffes, and Dr. Robert Braun for being on my dissertation defense committee. I would also like to thank Dr. Raghupathy Sivakumar, Dr. Chuanyi Ji, and Dr. Paul G. Steffes for being on my dissertation proposal committee. Their invaluable comments have helped me to achieve a solid research path towards this thesis.

I would like to acknowledge the members of the Broadband and Wireless Networking Laboratory (BWN-Lab) due to the excellent atmosphere they created. I am especially thankful to Eylem Ekici, Tuna Tuğcu, and Mehmet C. Vuran for their friendship and support.

Finally, I thank my parents, my brother, my sister, and my love for their constant support, love, encouragement, and sacrifices.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

AIMD	Additive Increase Multiplicative Decrease
ARC	Analytical Rate Control
ARQ	Automatic Repeat reQuest
ATI	Available Timeline Information
ATL	Adaptive Transport Layer
ATM	Asynchronous Transfer Mode
AWGN	Additive White Gaussian Noise
BER	Bit Error Rate
BM	Blackout Mitigation
CCSDS	Consultative Committee for Space Data Systems
DTN	Delay Tolerant Networking
ESRT	Event-to-Sink Reliable Transport
FEC	Forward Error Correction
GEO	Geosynchronous Earth Orbit
IETF	Internet Engineering Task Force
IMT-2000	International Mobile Telecommunication System - 2000
IP	Internet Protocol
IPN	InterPlaNetary
IRS	InterPlaNetary Relay Station
ITP	Integrated Transmission Protocol
LFC	Local Flow Control
MAC	Medium Access Control
NASA	National Aeronautics and Space Administration

NGI	Next Generation Internet
NGWI	Next Generation Wireless Internet
QoS	Quality of Service
RCS	Rate Control Scheme
RTT	Round Trip Time
RTP	Real-time Transport Protocol
RTCP	Real-time Transmission Control Protocol
SACK	Selective Acknowledgment
TCP	Transmission Control Protocol
TFRC	TCP-Friendly Rate Control
TOS	Type of Service
TP-Planet	Transport Protocol-Planet
UDP	User Datagram Protocol
UMTS	Universal Mobile Telecommunications System
WLAN	Wireless Local Area Network
WSN	Wireless Sensor Network
WWAN	Wireless Wide Area Network

SUMMARY

The revolutionary advances in the wireless communication technologies are inspiring the researchers to envision the next generation wireless networking architectures, i.e., Next Generation Wireless Internet (NGWI), InterPlaNetary (IPN) Internet, and Wireless Sensor Networks (WSN). There exist significant technological challenges for the realization of these envisioned next generation network architectures. NGWI will be the convergence of the Internet and currently developed heterogeneous wireless architectures, which have diverse characteristics and hence pose different sets of research challenges, to achieve *anywhere, anytime* seamless service to the mobile users. Similarly, the unique characteristics and challenges posed by deep space communications call for novel networking protocols to realize the IPN Internet objective. Furthermore, in order to realize the potential gains of WSN, it is imperative that communication challenges imposed by resource constraints of sensor nodes must be efficiently addressed with novel solutions tailored to the WSN paradigm. The objective of this research is to develop new advanced transport protocols for reliable data transport and real-time multimedia delivery in the next generation heterogeneous wireless network architectures. More specifically, the analytical rate control (ARC) protocol for real-time multimedia delivery is first proposed for wired/wireless hybrid networks. Next, a new rate control scheme (RCS) is proposed to achieve high throughput performance and fairness for real-time multimedia traffic over the satellite links. The unified adaptive transport layer (ATL) suite and its protocols for both reliable data transport (TCP-ATL) and real-time multimedia delivery (RCP-ATL) are introduced for the NGWI. A new reliable transport protocol for data transport in the IPN Internet (TP-Planet) is then proposed to address the unique challenges of the IPN Internet backbone links. A new integrated tranmission protocol (ITP) is then proposed for reliable data transport over multihop IPN Internet paths. Finally, the event-to-sink reliable transport (ESRT) protocol is proposed to achieve reliable event transport with minimum energy expenditure in WSN.

CHAPTER I

INTRODUCTION

The revolutionary advances in the wireless communication technologies are enabling the realization of wide range of heterogeneous wireless systems from cellular wireless networks to satellite networks. This technological evolution is further inspiring the researchers to envision the next generation wireless networking architectures, i.e., Next Generation Wireless Internet (NGWI), InterPlaNetary (IPN) Internet, and Wireless Sensor Networks (WSN). All of these next generation heterogeneous wireless network architectures have distinct communication paradigms and challenges imposed by their unique characteristics and the objectives involved in their design and deployment.

1.1 Next Generation Wireless Internet

The spectacular evolution of the wireless network technologies has yielded several generations of wireless systems referred by nG, $n = 1, 2, 3, 4$. Currently proposed third generation (3G) wireless systems such as Universal Mobile Telecommunication System (UMTS) and International Mobile Telecommunication System 2000 (IMT-2000) can provide up to 2Mbps data rates along with the multimedia support [90]. These developments in the wireless networking technologies and the drastic increase in their usage have further dictated the integration of wired and wireless domains. Consequently, the Next Generation Wireless Internet (NGWI), as also referred as 4G wireless networks which is currently in the design stage, will emerge as the convergence of next generation wireless systems and Internet with the objective of *anywhere, anytime* seamless service to the mobile users.

A typical proposed NGWI architecture is illustrated in Figure 1. In this architecture, an NG wireless terminal can roam between the diverse set of wireless architectures, i.e., WLAN, 3G cellular, and satellite network, as shown in Fig 1. Along with this architectural diversity, NGWI is also expected to provide the mobile users with a wide range of services

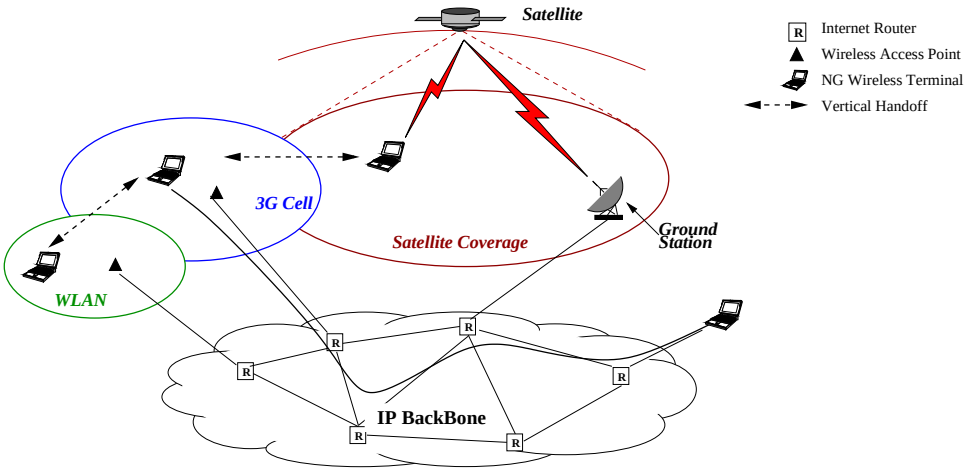

Figure 1: A typical proposed Next Generation Wireless Internet architecture including WLAN, 3G, and satellite networks converged with the Next Generation Internet backbone.

from high rate data traffic to real-time multimedia traffic with a certain QoS level. These objectives pose several challenges for the realization of the NGWI as follows:

- **Heterogeneous Wireless Architectures:** The wireless systems that will be part of the NGWI have different characteristics summarized as follows:
  - *Different access delays* are experienced in heterogeneous wireless architectures of NGWI, i.e., negligble delay in WLAN, several hundred milliseconds in 3G links due to the extensive physical layer processing, and to 270 ms in satellite links [10].
  - *Link errors rates significantly vary* from very low levels in near-wireless environments such as WLAN, 3G pico-cells to higher than 1% in macro-cellular environments and satellite networks [106].
  - *Mobility pattern* affects the probability of link outage and resultant packet loss rate, which drastically vary according to the wireless architecture.
- **Heterogeneous Service Demands:** The services that will be provided by the NGWI vary from high rate reliable data to real-time multimedia such as live video streaming.

These challenges need to be addressed to realize the NGWI objective. The architectural heterogeneities described above need to be handled dynamically while the mobile user migrates during connection period. The diversity in the demanded services necessitates both the reliable data transport functionality and the rate control protocol for time-constrained multimedia delivery.

1.2 InterPlaNetary Internet

The developments in the space technologies are enabling the realization of deep space scientific missions such as Mars exploration. The vision of future space exploration includes missions to deep space that require communication among planets, moons, satellites, asteroids, robotic spacecrafts, and crewed vehicles. These missions produce significant amount of scientific data to be delivered to the Earth. In addition, these missions require autonomous space data delivery at high data rates, interactivity among the in-space instruments, security of operations, and seamless inter-operability between in-space entities.

For successful transfer of scientific data and reliable navigational communications, NASA enterprises have outlined significant challenges for development of next-generation space network architectures. The next step in the design and development of deep space networks is expected to be the Internet of the deep space planetary networks and defined as the InterPlaNetary (IPN) Internet [114].

The InterPlaNetary Internet is envisioned to provide communication services for scientific data delivery and navigation services for the explorer spacecrafts and orbiters of the future deep space missions [23]. Many of these future planetary missions, which will be performed by the international space organizations such as NASA and European Space Agency (ESA), have already been scheduled for the next decade [110]. All of these future space missions have a common objective of scientific data acquisition and delivery, which are also the main possible applications of the InterPlaNetary Internet.

A proposed InterPlaNetary Internet communication architecture is shown in Figure 2 [12]. The architectural elements of the proposed infrastructure can be summarized as follows:

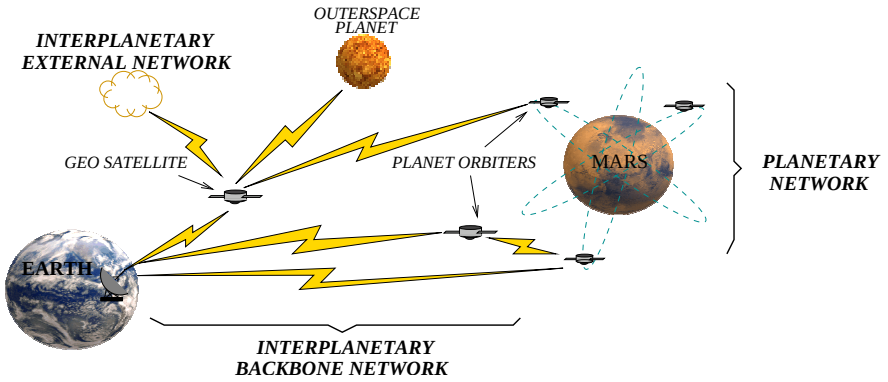

Figure 2: The InterPlaNetary Internet architecture.

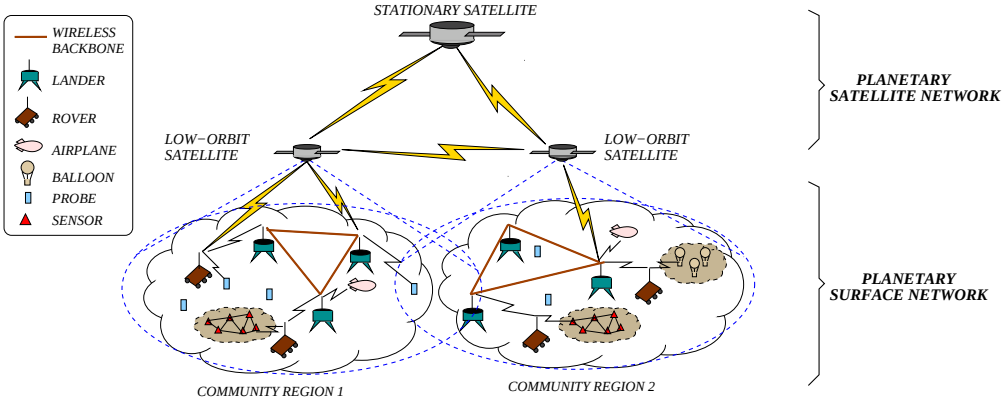

Figure 3: The PlaNetary Network architecture.

- **InterPlaNetary Backbone Network:** It provides a common infrastructure for communications among the Earth, outer-space planets, moons, satellite, and intermediate relay stations etc. It includes the data links between elements with long-haul capabilities.

- **InterPlaNetary External Network:** It consists of spacecrafts flying in groups in deep space between planets, clusters of sensor nodes, and groups of space stations.

- **PlaNetary Network:** It is composed of *PlaNetary Satellite Network* and *PlaNetary Surface Network*. This architecture can be implemented at any outer-space planet, providing interconnection and cooperation among the satellites and surface elements on a planet.

- **PlaNetary Satellite Network:** It is composed of multiple-layer satellites circling the planets as shown in Figure 3 and provides intermediary caching and relay service between the Earth and the planet, relay service between the in-situ mission elements, and location management of PlaNetary Surface Networks.
- **PlaNetary Surface Network:** It provides the communication links between high power surface elements, such as rovers and landers as shown in Figure 3, which have the capability to connect with satellites. They also provide a power-stable wireless backbone in the planet. Moreover, PlaNetary Surface Network includes surface elements that cannot communicate with satellites directly. These elements are often organized in clusters and spread out in an ad hoc manner, e.g., sensor nodes and balloons as illustrated in Figure 3.

It is clear that the InterPlaNetary Internet is expected to extend the current space communications capabilities to a point where the boundaries between the terrestrial and space communications become transparent. The experience obtained, thus far, from the space missions help to understand the unique challenges posed by the deep space communication environments. For example, the current communication infrastructure deployed for the NASA's Deep Space Network (DSN) [109] provides significant research and implementation experience which also constitutes the basis to step up the development of the required technologies for next generation deep space communications and hence the InterPlaNetary Internet. However, there exist significantly challenging and unique characteristics of the deep space networking paradigm that need to be addressed for this objective as follows:

- *Extremely long and variable propagation delays.*
- *Asymmetrical forward and reverse link capacities.*
- *High link error rates for radio-frequency (RF) communication channels.*
- *Intermittent link connectivity.*
- *Lack of fixed communication infrastructure.*

- *Effects of planetary distances on the signal strength and the protocol design.*
- *Power, mass, size, and cost constraints for communication hardware and protocol design.*
- *Backward compatibility requirement due to high cost involved in deployment and launching processes.*

These characteristics lead to different research challenges and hence necessitate different approaches and protocol designs at each of the networking layers for the InterPlaNetary Internet. Although some of these challenges are also encountered in the terrestrial wireless networking domain, most of them are unique to deep space environments and they further amplify the effects of those other similar factors. Consequently, the realization of the InterPlaNetary Internet depends on how effectively these challenges are addressed.

1.3 Wireless Sensor Networks

The technological advances in the micro-electro-mechanical systems (MEMS) and the wireless communications are enabling the deployment of wireless sensor networks (WSN) which may provide sensing capabilities in space and time that surpass the traditional sensing approaches. WSNs can be deployed in the following two ways [62]:

- Sensors can be positioned far from the actual *phenomenon*, i.e., something known by sense perception. In this approach, large sensors that use some complex techniques to distinguish the targets from environmental noise are required.
- Several sensors that perform only sensing can be deployed. The positions of the sensors and communications topology are carefully engineered. They transmit time series of the sensed phenomenon to the central nodes where computations are performed and data are fused.

While there are similarities such as communication through shared wireless medium and lack of networking infrastructure; there exist clear differences between the wireless sensor networks and traditional wireless ad hoc networks. The main distinction is that

unlike wireless ad hoc networks mostly constructed for communication purposes, the sensor networks are deployed to achieve a specific application objective via collaborative effort of numerous sensor nodes. With this respect, the number of sensor nodes and node density in a sensor network can be several orders of magnitude higher than the nodes in an ad-hoc network. For this reason, sensor nodes also do not have global *identification* (ID) because of the large amount of overhead and large number of sensor nodes. Moreover, the sensor nodes have much more constrained energy, communication, processing, and memory capabilities as compared to traditional wireless ad hoc devices.

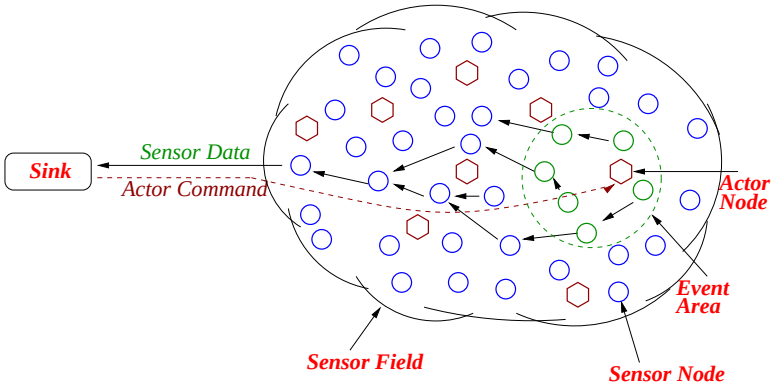

Figure 4: A typical wireless sensor network architecture.

In general, wireless sensor networks (WSN) are event based systems that rely on the collective effort of densely deployed several microsensor nodes which continuously observe physical phenomenon. Hence, the main objective of the WSN is to reliably detect/estimate event features from the collective information provided by sensor nodes as illustrated in Figure 4. The energy and hence processing constraints of small wireless sensor nodes are overcome by this collective sensing notion which is realized via their networked deployment. Based on the event features communicated to the sink, the actors in the event vicinity may be invoked to act on the detected phenomenon.

Due to its potential advantages of collective, efficient, and accurate sensing, the wireless sensor nodes are envisioned to be exploited for many applications, e.g., location tracking and chemical detection in areas not easily accessible. Since sensing applications generate a large quantity of data, these data may be fused or aggregated together to lower the energy

consumption. The sensor nodes use their processing abilities to locally carry out simple computations and transmit only the required and partially processed data. In essence, wireless sensor networks with these capabilities may provide the end users with intelligence and a better understanding of the environment. Furthermore, the wireless sensor networks (WSN) may also be integrated with the next generation wireless Internet in order to enable end users to interact with sensing/monitoring applications and to reach the collected information from the wireless sensor field. In the future, the wireless sensor networks may be an integral part of our lives, more so than the present-day personal computers. Nevertheless, in order to realize these potential gains of wireless sensor networks, it is imperative that communication challenges imposed by resource constraints of the wireless sensor nodes must be efficiently addressed with novel solutions tailored to the WSN paradigm.

1.4 Research Objectives and Solutions

In order to realize these next generation heterogeneous wireless network architectures introduced in Section 1.1, Section 1.2, and Section 1.3, the communication challenges posed by each of these environments must be effectively addressed. In this thesis, new advanced transport protocols are developed for reliable data transport and real-time multimedia delivery in the next generation heterogeneous wireless network architectures, i.e., Next Generation Wireless Internet, InterPlaNetary Internet, and Wireless Sensor Networks. The following six areas are investigated under this research and each of them is described in the following subsections:

1. Analytical Rate Control in Hybrid Wired/Wireless Networks
2. Multimedia Rate Control in Satellite Networks
3. Adaptive Transport in NGWI
4. Reliable Data Transport in IPN Internet
5. Integrated Transmission in IPN Internet
6. Reliable Event Transport in WSN

1.4.1 Analytical Rate Control in Hybrid Wired/Wireless Networks

The need of rate control for real-time flows in Internet has been subject of many papers in the literature [18], [32], [88], [87], [52]. The main objective of these schemes is to perform rate control for the real-time multimedia streaming in order to maintain fair share of the network resources and hence avoid congestion collapse of the Internet. However, most of the effort has been put on the rate control research for wireline networks rather than wireless networks. These solutions assume error-free communication path, which leads to recognition of wireless channel error related packet losses as a congestion indication resulting in severe performance degradation over the hybrid wired/wireless paths in the NGWI.

In this thesis, a new analytical rate control (ARC) scheme is presented for real-time multimedia flows over hybrid wired/wireless communication paths. ARC is an end-to-end connectionless analytical rate control scheme, which produces TCP-Friendly flows with high throughput over wireless links. It is an equation-based rate control protocol whose control equation is obtained via modeling TCP behavior over lossy links by excluding its shortcomings. In order to obtain such equation for rate control over wireless networks, an end-to-end path model is developed based on two-state Markov Chain known as Gilbert's Model [54]. From the established model, the driving equation for the ARC scheme is derived based on the desired behavior of a TCP source as a function of the state probabilities of the path model and the round-trip time (RTT). It is then used to explicitly set the data transmission rate for the multimedia flows. Performance evaluation via simulation experiments reveals that ARC achieves high throughput and meets multimedia traffic expectations without violating good citizenship rules for the shared Internet.

1.4.2 Multimedia Rate Control in Satellite Networks

Recently, much research work has been done to develop rate control protocols for real-time multimedia applications in the terrestrial wireline networks [18], [26], [32], [52], [63], [69], [87], [88], [100]. The common objective of these proposed solutions is to enable the real-time multimedia applications to respond to the network congestion by controlling their bandwidth share such that the link resources are fairly shared. Main drawback of them, however,

is that these solutions are developed for wired links which are assumed to have negligible errors and hence they cannot be directly applied to the links with high bit error rates and high bandwidth-delay products. The performance degradation due to wireless channel errors is further amplified and the problem becomes much more complicated in the paths with high-bandwidth delay products such as satellite links which have high propagation delay up to 270 ms [10]. This is mainly because the recovery from such an inaccurate congestion response of rate throttle takes time proportional to the bandwidth-delay product of the path.

In this thesis, a new Rate Control Scheme (RCS) is introduced for real-time interactive applications in networks with high bandwidth-delay products and high bit error rates such as satellite networks. RCS is based on the concept of using dummy packets to probe the availability of network resources. Dummy packets are treated as low priority packets and consequently they do not affect the throughput of actual data traffic. Therefore, RCS requires all the routers in the connection path to support some priority policy. A new algorithm is also proposed to improve the robustness of the RCS to temporal signal loss conditions. The delay-bound considerations for real-time traffic sources using RCS rate control scheme are also investigated. Simulation experiments show that in environments with high bandwidth-delay products and high bit error rates, RCS achieves high throughput performance without penalizing TCP connections.

1.4.3 Adaptive Transport in NGWI

Despite the extensive research in transport protocols to address the challenges in wireless domain [99], [31], [20], [17], [10], [11], [57], most of them address these challenges only for a specific wireless environment and hence do not solve the problems pertaining to the NGWI which will be a convergence of heterogeneous wireless network architectures as desribed in Section 1.1. For the NGWI architecture in which the mobile user migrates between the different environments as illustrated in Figure 1, none of these protocols will suffice. Similarly, although there exists a significant amount of research for rate control schemes in this context [88], [73], [32], [87], [18], [4], [3], most of them are proposed for wired

networks and the others are developed for specific wireless environments, e.g., [3] for hybrid wired/wireless networks and [4] for satellite networks. Furthermore, including each of the existing transport protocols tailored for a specific architecture in a single transport layer is not a practical solution due to the processing and memory constraints of wireless terminals. Hence, there exists a need for a unified transport layer to handle both architectural diversity and heterogeneous application requirements of the NGWI.

In this thesis, a unified adaptive transport layer (ATL) suite is introduced for NGWI which incorporates a new adaptive transport protocol, TCP-ATL, for reliable data transport; and a new adaptive rate control protocol, RCP-ATL, for multimedia delivery in the NGWI. According to the requested service type, i.e., reliable data or multimedia, ATL selects the appropriate protocol. Both, TCP-ATL and RCP-ATL, deploy a new adaptive congestion control method that dynamically adjusts the protocol configurations according to the current wireless network paradigms depending where the mobile user currently resides. Hence, the unified adaptive ATL protocol suite achieves high throughput performance in all of underlying heterogeneous wireless architectures, i.e., WLANs, micro, macro or satellite environments. Moreover, the developed adaptive congestion control explicitly takes fairness into consideration. Performance evaluation via simulation experiments reveals that the ATL protocol suite addresses the challenges posed by the NGWI and significantly improves the performance for reliable data and multimedia transport in NGWI.

1.4.4 Reliable Data Transport in IPN Internet

The space exploration missions are crucial for acquisition of information about the space and the universe. The entire success of a mission is directly related to the satisfaction of its communications needs. For this goal, the challenges posed by the InterPlaNetary Internet need to be addressed. However, in [2], the existing reliable transport protocols [10], [28], [31], [51], [65], [66], [81], [80] have been shown to achieve very poor performance in deep space communication networks mainly because of the inefficiency caused by their end-to-end window-based congestion control mechanisms due to very high propagation delay and link errors. Although there exists transport protocol solutions proposed for satellite links with

high delays and high bit error rates [10], these solutions also cannot be directly applied to IPN Backbone Network because of the extremely high propagation delay and the other additional challenges mentioned above. Space Communications Protocol Standards-Transport Protocol (SCPS-TP) [38] is a set of TCP extensions that are basically a combination of existing TCP protocols with some modifications and extensions, which are shown to be inadequate in addressing the challenges in IPN Backbone Network.

In this thesis, a reliable transport protocol, TP-Planet, is presented for data traffic in the InterPlaNetary Internet. The objective of TP-Planet is to address the challenges and to achieve high throughput performance and reliable data transmission on deep space links of the InterPlaNetary Backbone Network. TP-Planet is the first reliable data transport solution for InterPlaNetary Internet. TP-Planet deploys a rate-based additive-increase multiplicative-decrease (AIMD) congestion control, whose AIMD parameters are tuned to help avoid throughput degradation. TP-Planet replaces the inefficient slow start algorithm with a novel Initial State algorithm which allows to capture link resources in a very fast and controlled manner. A new congestion detection and control mechanism is developed, which decouples congestion decision from single packet loss in order to avoid the erroneous congestion decisions due to high link errors. In order to reduce the effects of blackout conditions on the throughput performance, TP-Planet incorporates Blackout State procedure into the protocol operation. Bandwidth asymmetry problem is addressed by the adoption of delayed SACK. Simulation experiments show that the TP-Planet significantly improves the throughput performance and addresses the challenges posed by the InterPlaNetary Backbone Network.

1.4.5 Integrated Transmission in IPN Internet

It has been shown in [2] that the existing reliable transport protocols significantly suffer from extremely high propagation delays and high bit errors in deep space communication networks because of the inefficiency caused by their end-to-end window-based congestion control mechanisms. In general, the end-to-end congestion control and reliability mechanisms are deemed to suffer from the adverse effects of the above challenges. The conventional

transport layer approaches with end-to-end congestion control and reliability mechanisms assume the existence of an end-to-end path from the source to the final destination. Such assumption may not be totally unreasonable for some IPN Internet paths with very few hops such as Earth-Mars communication network despite occasional blackout situations. In fact, a novel end-to-end reliable data transport protocol, TP-Planet presented in Chapter 5, efficiently addresses the above challenges and significantly improve the throughput performance in Mars exploration communication architecture [6]. However, intermittent connectivity is one of the main characteristics of the IPN Internet and hence should be considered as part of any connection rather than an occasional and extreme network condition. In this case, end-to-end transport layer protocols experience significant network inefficiency due to frequent retransmissions performed all the way from the source to the destination. This problem is further amplified in the IPN Backbone Network paths spanning several IPN hops, which increases the probability of experiencing blackouts in a given connection period. To address these problems, hop-by-hop bundle transport approach [95] is currently being developed for the IPN Internet. However, there is no reliable data transport mechanism to provide hop-by-hop congestion control and reliability functionalities.

In this thesis, a new Integrated Transmission Protocol (ITP) is presented for reliable data transport in the IPN Internet. ITP is a unified transmission protocol solution for hop-by-hop congestion control and reliability mechanisms specifically tailored for IPN Internet paths with intermittent connectivity. The objective of ITP is to address the challenges and to achieve high performance reliable data transport on deep space links of the IPN Internet. Exploiting the hop-by-hop nature of the connections, ITP unifies the common functionalities of the conventional transport and link layers. ITP deploys a new rate-based hop-by-hop local flow control (LFC) mechanism; which exploits the local resource availability and traffic information at the receiver in order to provide explicit available bandwidth feedback to the sender. LFC mechanism decouples congestion decision from reliability to avoid the erroneous congestion decisions due to high link errors and avoids the delayed-feedback problem. ITP incorporates the selective-acknowledgment (SACK) based Automatic Repeat reQuest (ARQ) to assure hop-by-hop local packet-level transport

reliability. To reduce the effects of blackout conditions on the performance, ITP includes the Blackout Mitigation (BM) procedure. The optimum packet size is analytically obtained to further improve the transmission efficiency over deep space links. Bandwidth asymmetry problem is addressed by the adoption of delayed SACK. Simulation experiments show that the ITP significantly improves the throughput performance and addresses the challenges posed by the IPN Internet.

1.4.6 Reliable Event Transport in WSN

Wireless sensor networks (WSN) are event based systems that rely on the collective effort of several microsensor nodes. Reliable event detection at the sink is based on collective information provided by source nodes and not on any individual report. Although there exist some solution proposals to address the transport layer problems in wireless sensor networks in the literature [118], [103], [117], these solutions either follow conventional reliability approaches involving in reliable transmission of individual data packet or do not provide reliability at all. While conventional end-to-end reliability definitions and solutions are inapplicable in the WSN regime and would only lead to a waste of scarce sensor resources, the absence of reliable transport altogether can seriously impair event detection. Hence, the WSN paradigm necessitates a collective *event-to-sink* reliability notion rather than the traditional end-to-end notion. Reliable transport in WSN has not been studied from this perspective before.

In this thesis, a new reliable transport scheme for WSN, the event-to-sink reliable transport (ESRT) protocol, is presented. ESRT is a novel transport solution developed to achieve reliable event detection in WSN with minimum energy expenditure. It includes a congestion control component that serves the dual purpose of achieving reliability and conserving energy. Importantly, the algorithms of ESRT mainly run on the sink, with minimal functionality required at resource constrained sensor nodes. ESRT protocol operation is determined by the current network state based on the reliability achieved and congestion condition in the network. If the event-to-sink reliability is lower than required, ESRT adjusts the reporting frequency of source nodes aggressively in order to reach the target reliability level as

soon as possible. If the reliability is higher than required, then ESRT reduces the reporting frequency conservatively in order to conserve energy while still maintaining reliability. This self-configuring nature of ESRT makes it robust to random, dynamic topology in WSN. Furthermore, ESRT can also accommodate multiple concurrent event occurrences in a wireless sensor field. Analytical performance evaluation and simulation results show that ESRT converges to the desired reliability with minimum energy expenditure, starting from any initial network state.

1.5 Thesis Outline

This thesis is organized as follows. Chapter 2 presents the analytical rate control (ARC) protocol developed for real-time multimedia delivery in wired/wireless hybrid networks. Chapter 3 introduces a new rate control scheme (RCS) is proposed to achieve high throughput performance and fairness for real-time multimedia traffic over the satellite links with high bit error rates and high bandwidth-delay products. Chapter 4 presents the unified adaptive transport layer (ATL) suite and its protocols for both reliable data transport and real-time multimedia delivery which deploy adaptive congestion control mechanism developed to address the heterogeneities posed by the NGWI. Chapter 5 describes a new reliable transport protocol (TP-Planet) for data transport in the InterPlaNetary Internet which addresses the unique challenges of the InterPlaNetary Internet backbone links and achieve high rate reliable data transport. The performance evaluation of the existing transport protocols over deep space communication networks is given in this chapter. Chapter 6 presents the new integrated transmission protocol (ITP) which unifies common transport and link layer functionalities to achieve efficient hop-by-hop flow control and reliability over the IPN Backbone Network links with intermittent connectivity. Chapter 7 introduces the event-to-sink reliable transport (ESRT) protocol which achieves reliable event transport with minimum energy expenditure and congestion avoidance in the wireless sensor networks. Finally, Chapter 8 summarizes the research results and suggests a number of problems for future investigation.

CHAPTER II

ANALYTICAL RATE CONTROL IN HYBRID WIRED / WIRELESS NETWORKS

In this chapter, a new analytical rate control (ARC) protocol for real-time multimedia traffic over wireless networks is presented. The objective of ARC is to achieve high throughput and multimedia support for real-time traffic flows while preserving fairness to the TCP sources sharing the same wired link resources over hybrid wired/wireless paths of the NGWI. The ARC protocol was first introduced in [3]. The existing related work in the literature is explored in Section 2.2. The end-to-end path model which constructs the physical layer abstract for the derivation of rate control equation is established in Section 2.3. In Section 2.4, based on the end-to-end path model, the desired behavior of a TCP source over lossy links is captured via renewal theory. In Section 2.5, the ARC protocol algorithm and its operation are introduced. The simulation experiment results and performance evaluation are discussed in Section 2.6.

2.1 Motivation

Higher sensitivity to time constraints such as delay and jitter, and more tolerance to packet losses are the characteristics of real-time multimedia traffic. These specifications also constitute the reason why there is no transmission rate control for multimedia applications. This fact would not lead to a problem, if the quality of service (QoS) oriented and reservation-based *DiffServ* [25] or *IntServ* [123] architectures were used. However, there will always exist a necessity of a rate control mechanism for real-time flows to avoid unfair resource sharing among competing sources and congestion collapse. The transmission rate control can be in co-operation with encoding process. Hence, the real-time multimedia encoder can then adapt its encoding rate in accord with the controlled fair share of network resources [119].

2.2 Related Work

The need of rate control for real-time flows in Internet has been subject of many papers in the literature [18], [32], [88], [87]. However, most of the effort has been put on the rate control research for wireline networks rather than wireless networks. The common feature of these proposals is that they follow the conservative rate halving behavior of TCP. This behavior is inherited from the assumption of reliable and almost error-free communication link presence. This leads to recognition of a single packet loss as a congestion indication, which may well not hold in error-prone wireless links. Therefore, TCP-like rate control for wireless networks leads to unnecessary rate throttle and hence severe performance degradation. One way to overcome this problem is to distinguish packet losses due to link errors from congestion. Although there are some proposed methods to achieve this [37], [84], [91], it is not easy to obtain high accuracy in determination of actual reason for packet loss to take proper action. In [106], Rate Control Scheme (RCS) has been proposed for real-time traffic in the networks with high bandwidth-delay products and high bit error rates. RCS significantly improves the throughput performance while maintaining fairness. However, its *dummy packet* based congestion control algorithm is specifically tailored to match the requirements of the typical satellite links with high propagation delay and may be inefficient for the wireless environments with low access delay such as WLAN and pico-cells.

In addition to additive-increase multiplicative-decrease (AIMD) rate control proposals in the literature, some research has also been performed on equation-based congestion control [52], [83], [115], [125]. Instead of responding to a single packet loss, equation-based congestion control uses a control equation to adjust data rate. TCP-Friendly Rate Control (TFRC) [52] is the most important equation-based congestion control proposal in the literature. The control equation of TFRC is the TCP response function [83] governing the steady-state transmission rate of TCP as a function of round-trip time, RTT, and loss event rate, p. The loss event rate is calculated at the receiver and sent to the sender. The source, having loss event rate and RTT, can adjust its data rate by using its control equation. Hence, TFRC achieves fairness and smooth rate change with equation-based congestion control.

The equation-based rate control can also be used in wireless networks to address the requirements of real-time multimedia transport and the characteristics of wireless links. However, the existing equation-based rate control schemes cannot be directly applied to the wireless environments. The fundamental reason is that the throughput equation in [83] models the steady-state TCP behavior over error-free wireline links. This approach is again based on the assumption that a packet loss is an indication of congestion and hence is directly related to the actual link resources. For instance, TFRC [52] uses loss event rate, which is calculated by considering packet loss event observed at receiver. In wireless scenario, this approach can not distinguish the packet loss reasons and the calculated loss event rate contains also the packet losses due to wireless link errors. Therefore, it may result in inaccurate rate determination and hence underutilization of the link resources.

There also exist some studies in the literature that model the TCP behavior over lossy wireless channels. For instance, a stochastic analytical model of TCP-Reno over lossy links is presented in [1]. In [72], TCP performance over networks with high bandwidth-delay products and random loss is analyzed. Nevertheless, the plain TCP behavior over wireless links is not desirable since TCP itself is not well-suited to lossy links. Therefore, the response function obtained from the throughput model of TCP over wireless links should not be the control equation to achieve efficient analytical rate control. The required throughput model should reflect the desired TCP behavior over lossy links, i.e.,

- TCP source should not throttle data rate in case of packet loss due to wireless link error,
- It should follow standard TCP rules otherwise.

Hence, what should be modeled is not the actual TCP behavior rather its desired behavior over wireless links. The throughput equation obtained out of such model can achieve throughput efficient and TCP-friendly rate control. Thus, it is necessary to obtain a new model, whose response function can be used to control the data rate of real-time flows over wireless links.

In this chapter, in order to address the issues introduced above, the analytical rate

control (ARC) scheme is presented for real-time multimedia flows over wireless links. Two key features of the analytical rate control approach construct the departure point:

- It decouples a single packet loss event from triggering rate control process.
- It achieves smooth variance in transmission rate, i.e., support for real-time multimedia transport.

ARC is an end-to-end connectionless analytical rate control scheme, which produces TCP-Friendly flows with high throughput over wireless links. It is an equation-based rate control protocol whose control equation is obtained via modeling TCP behavior over lossy links by excluding its shortcomings. In order to obtain such equation for rate control over wireless networks, an end-to-end path model based on two-state Markov Chain known as Gilbert's Model [54] is first established. From the established model, the driving equation for the analytical rate control scheme is derived based on the desired behavior of a TCP source. The control equation is a function of the state probabilities of the path model and the round-trip time (RTT). The overall ARC protocol operation consists of two sequential periods, i.e., *Probe period* and *Steady period*. Since no link information is available in the beginning of the connection, the analytical rate control cannot be invoked. During this period, the ARC source calls the `Probe()` algorithm in order to determine the initial data transmission rate. At the end of this period, the source goes into *steady period* calling the `Steady()` algorithm. During the steady period, the data transmission rate S is controlled by using (21) as the required link information becomes available.

2.3 Communication Path Model

The end-to-end communication path is a concatenation of wireless and wired links. Hence, in order to obtain an accurate throughput model, it is necessary to capture channel behavior for the overall communication path. In the next subsections, the models for wireless channel and wired link are setup individually, and then the channel model for the end-to-end path is established.

2.3.1 Wireless Channel Model

Existing studies in the literature justify that first-order Markov chain such as two-state Markov Model provides adequate approximation to error process of Rayleigh fading channel behavior [129], [121]. Hence, the wireless link part of the end-to-end connection path is modeled with two-state discrete Markov chain known as Gilbert's Model [54], [105], [120] as shown in Figure 5.

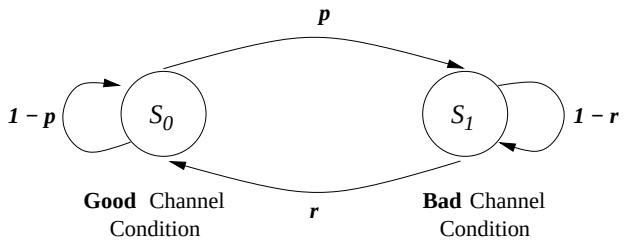

Figure 5: Two-state Markov chain for packet loss process model for wireless channel.

Two-state Markov chain can capture the bursty nature of the packet errors in fading channels. The model has two states as S_0 and S_1 representing *good* and *bad* channel conditions, respectively. When the state is S_0, i.e., good state, a packet is transmitted successfully. The transmission fails if the channel is in bad state, i.e., S_1. Success or failure of the transmission only depends on the current state. p and r are the state transition probabilities. The transition between two states occurs at each packet instant. For $P(S_0) + P(S_1) = 1$ being steady-state state probabilities, the state transition matrix is given by

$$\mathbf{P_S} = \begin{bmatrix} 1-p & p \\ r & 1-r \end{bmatrix} \tag{1}$$

The state transition probabilities depend on error characteristics of the physical layer. The transition probabilities can be calculated from the channel characteristics such as packet error rate (PER). The packet error rate (PER) of the wireless link is equal to the state probability $P(S_1)$ and depends on the transition probabilities [14]. Similarly, the probability of successful delivery of the packets over the wireless channel is, therefore, equal to the state probability $P(S_0)$.

2.3.2 Wired Link Model

Recent works on the measurements of unicast and multicast packet loss in the Internet have shown that the error process in the Internet can also be modeled with two-state Markov Chain [13], [27]. The state diagram with two states representing good and bad conditions, I_0 and I_1, is shown in Figure 6.

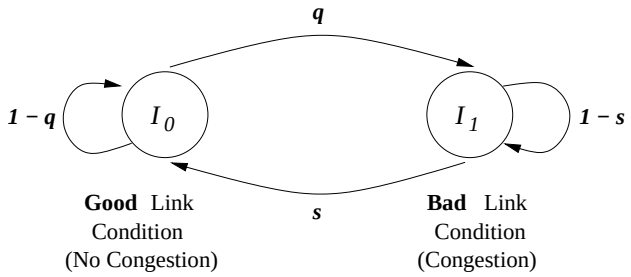

Figure 6: Two-state Markov chain for packet loss process model for wired link.

Here, the bad wired link condition represents congestion situations. Hence, a packet is lost due to congestion, when the state is I_1. Packet is transmitted successfully otherwise. q and s are the state transition probabilities. For $P(I_0) + P(I_1) = 1$ being steady-state state probabilities, the state transition matrix is given as follows

$$\mathbf{P_I} = \begin{bmatrix} 1-q & q \\ s & 1-s \end{bmatrix} \tag{2}$$

As in the wireless channel model, the packet loss rate due to congestion is equal to the state probability $P(I_1)$; and the probability of successful transmission over the wired link is, therefore, equal to the state probability $P(I_0)$.

2.3.3 End-to-End Path Model

Since the communication path consists of both wireless and wired parts, the model for the end-to-end path should capture behaviors of both portions of the path. Having individual packet error process models, the overall path model can be constructed as a concatenation of two models established in previous subsections. The resultant four-state chain representing end-to-end path is shown in Figure 7.

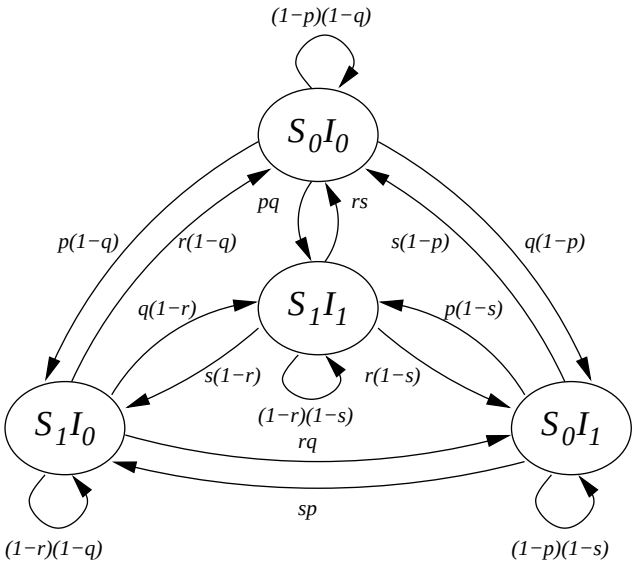

Figure 7: Four-state Markov chain for packet loss in the end-to-end path including wireless and wired links.

In this model, a transition occurs between good (S_0I_0) and three bad states corresponding to bad wireless channel condition (S_1I_0), bad wired link condition (S_0I_1), and both (S_1I_1). S_1I_0 and S_0I_1 states represent only wireless link error and congestion conditions, respectively. When the state is S_0I_0, packet transmission is successful. Transmission fails if the state is in one of the bad states. Success or failure of the packet transmission solely depends on the current state. For $P(S_0I_0) + P(S_1I_0) + P(S_0I_1) + P(S_1I_1) = 1$ being steady-state state probabilities, the state transition probability matrix is given as

$$\mathbf{P_{SI}}= \begin{bmatrix} (1-p)(1-q) & p(1-q) & q(1-p) & pq \\ r(1-q) & (1-r)(1-q) & rq & q(1-r) \\ s(1-p) & sp & (1-s)(1-p) & p(1-s) \\ rs & s(1-r) & r(1-s) & (1-s)(1-r) \end{bmatrix} \tag{3}$$

In the four-state Markov chain shown in Figure 7 representing the packet loss process for the end-to-end path, the total probability of packet loss, π, can be calculated by

$$\begin{aligned} \pi &= 1 - P(S_0I_0) \\ &= P(S_0I_1) + P(S_1I_0) + P(S_1I_1) \end{aligned} \tag{4}$$

where $P(S_0I_1)$ is the probability that wireless link is error-free and congestion exists; $P(S_1I_0)$ is the probability that wireless link is error-prone and no congestion exists; and $P(S_1I_1)$ is the probability that wireless link is error-prone and congestion exists. Since the total packet loss probability is the sum of the probability that packet loss is due to congestion and the probability that packet loss is due to wireless error; by rearranging (4), π can also be expressed as

$$\pi = 1 - [1 - P(S_1)]\,[1 - P(I_1)] \tag{5}$$

where $P(S_1)$ and $P(I_1)$ can be calculated by

$$P(S_1) = P(S_1I_0) + P(S_1I_1) \tag{6}$$

$$P(I_1) = P(S_0I_1) + P(S_1I_1) \tag{7}$$

Recall that $P(S_1)$ is the probability of being in the bad wireless channel state. On the other hand, the estimation of the state probability of $P(S_1)$ is a challenging task and there exists studies in the literature on the accurate estimation of the communication channel state probabilities and the packet loss classification techniques [74], [89]. For this reason, ARC protocol operation does not involve in direct estimation of $P(S_1)$ rather it uses the measured packet loss rate due to wireless link at the link layer as an approximate to the state probability of the bad wireless channel. The details of the ARC protocol operation are presented in Section 2.5.

Note that the packet loss rate at the wireless link and π are both measurable quantities. Therefore, instead of calculating transition probabilities, i.e., p, r, q, and s, the state probabilities of the end-to-end path model are used in developing the analytical rate control scheme as presented in Section 2.4.

2.4 Model For Desired TCP Behavior Over Wireless Links

Having established the model for the end-to-end communication path, the next step for the analytical rate control protocol is to derive the governing equation. The throughput equations used by the existing equation-based congestion control algorithms are based on

the TCP behavior over reliable wired links [52], [83]. Therefore, they are not directly applicable to the wireless environments, since the TCP has poor performance in the error-prone wireless links. Thus, a new model which does not represent the classical TCP behavior over wireless links needs to be obtained. The new model should, instead, capture the steady-state TCP behavior by excluding its drawbacks over lossy links. Therefore, the throughput model should also be consistent with the underlying channel model obtained for end-to-end path in the previous section.

Using the model that captures the actual TCP behavior over wireless links is not also appropriate for this purpose. The models for the TCP behavior over lossy links do exist in literature [1], [72] and provide us with a better understanding of TCP's shortcomings. However, the use of these models to obtain a response function for equation-based rate control leads to experience the same throughput degradation with the TCP over wireless links. Therefore, a model for the desired TCP behavior over lossy links which excludes its drawbacks needs to be obtained.

Most of the studies for TCP over wireless links aim to avoid TCP throughput degradation due to unnecessary rate decrease in case of link error losses by two major approaches. They either try to hide wireless link losses from TCP source, or try to distinguish the loss reason and then act accordingly [17]. The former approach requires split, or indirect solutions most of which are not scalable and do not obey end-to-end semantics of congestion control. The latter method needs very accurate and fast way of distinguishing loss reason. What is common in these studies is that all aim to make TCP source behave in a desired way so that TCP source over lossy link should

- perform no rate change due to wireless link error packet loss,
- decrease rate when packet loss due to congestion occurs.

Therefore, a throughput model for such desired TCP behavior on top of the developed end-to-end path model is obtained. The resulting equation will then be used as the data rate controller for real-time multimedia traffic sources in the wireless networks. Note also

that such rate control policy is suitable for the real-time multimedia flows which are loss-tolerant to a certain extent and have strict time constraints. Since the reliable transmission is of second importance due to the loss-tolerant nature of the real-time multimedia flows; it is better to keep transmitting time-sensitive packets with a controlled rate using a rate control protocol instead of backing off in case of wireless link errors.

Assuming that application always has data to send, TCP flow starting at $t = 0$ transmits $X(t)$ packets and achieves $T(t) = X(t)/t$ throughput in t time period. Therefore, the steady-state throughput, T, for such flow is

$$T = \lim_{t \to \infty} \frac{\overline{X}(t)}{t} \tag{8}$$

where $\overline{X}(t)$ is the expectation of the total number of packets transmitted in $[0, t]$ for $t \to \infty$.

The time-dependency of the congestion window size, W, is shown in Figure 8, where the packet loss events are due to either congestion loss (CL) or wireless link error (WL). Time-out events are not considered in this model. For packet loss rates up to 0.1, the model including only the triple-duplicate events is shown to be quite adequate approximation to the actual behavior in [83]. Furthermore, the goal is to capture desired source behavior over established path model and use its response function as a driving equation for rate control, rather than to obtain most accurate analytical model for TCP source behavior over reliable links as in [83]. As it is in Figure 8, there is no rate decrease at WL event arrivals but there is at CL events. This is consistent with the desired TCP behavior.

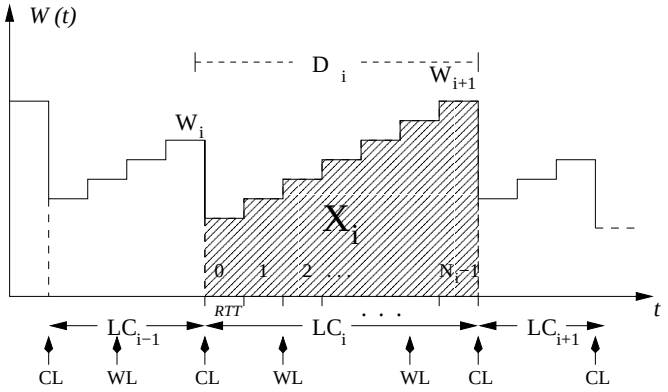

Figure 8: Steady state desired behavior of a TCP source in wireless link.

Each loss-free period, i.e., loss cycle LC_i, starts and ends with CL event arrival. W_i is

the congestion window size at the beginning of each loss cycle just before the rate halving occurs. For the total number of packets transmitted X_i in the duration of D_i, the throughput achieved during LC_i is $T_i = X_i/D_i$. For each LC_i ending with X_i number of packets sent, the evolution of congestion window W_i can be assumed to be Markov Regenerative Process [75] with rewards X_i. Hence, the asymptotic throughput from renewal theorem is expressed by

$$T = \frac{\overline{X}}{\overline{D}}, \tag{9}$$

where $\overline{X}$ and $\overline{D}$ are the expectations needed to obtain for X_i and D_i, respectively.

The congestion window at the j^{th} RTT, $W_i[j]$, in a loss cycle i can be expressed as follows

$$W_i[j] = \frac{W_i}{2} + j \tag{10}$$

N_i is the number of RTTs in one loss cycle period. Thus, for $N_i - 1$ is the RTT index in which the next congestion loss event occurs, W_{i+1} is the congestion window at the end of the i^{th} loss cycle and is equal to $W_i/2 + N_i$. Hence, the expectation of i.i.d random variable W denoting congestion window size in LC_i can be expressed as

$$\overline{W} = 2\overline{N} \tag{11}$$

The total number of packets X_i sent in the loss cycle LC_i lasting $N_i \cdot RTT$ duration can be calculated as follows

$$\begin{aligned} X_i &= \sum_{j=0}^{N_i-1} \left(\frac{W_i}{2} + j\right) \\ &= \frac{1}{2}\left(W_{i+1} + \frac{W_i}{2} - 1\right) N_i \end{aligned} \tag{12}$$

Consequently, for mutually independent random variables of N_i and W_i, the mean of random variable X_i can be expressed as

$$\overline{X} = \frac{1}{2}(3\overline{N} - 1)\overline{N} \tag{13}$$

In a loss cycle LC_i, n_i packets are transmitted before the congestion loss occurs. Since a packet loss is detected in one RTT period, $W_i[N_i - 1]$ more packets are sent after the

packet loss due to congestion occurs. Therefore, the total number of packet transmitted in LC_i can also be expressed as

$$X_i = n_i + W_i[N_i - 1] \tag{14}$$

The random variable denoting the number of packets transmitted n_i is geometrically distributed with the unconditional probability of packet loss due to congestion $P(I_1)$. Hence, the probability that n packets are transmitted (including the lost one) is

$$P[n_i = n] \quad = \quad (1 - P(I_1))^{n-1} P(I_1) \tag{15}$$

Let π be the total packet loss probability and ω be the probability of packet loss due to wireless link errors. Recall that $\omega = P(S_1)$. Hence, it follows from (4), (5), (6), and (7) that $P(I_1)$ can be calculated as

$$P(I_1) \quad = \quad \frac{\pi - \omega}{1 - \omega}\,, \tag{16}$$

where π is the total packet loss probability, and ω is the probability of packet loss due to wireless link errors.

Hence, n_i gives the number of packets transmitted until a packet loss occurs by congestion which leads to rate throttle, rather than the wireless link error. Therefore, the mean of the total number of packets transmitted X_i can also be expressed as

$$\overline{X} = \overline{n} + \overline{W}\,, \tag{17}$$

where the mean of random variable n_i can be obtained by using (15) and (16) as follows

$$\begin{aligned} \overline{n} \quad &= \quad \sum_k k P[n_i = k] \\ &= \quad \frac{1 - \omega}{\pi - \omega} \end{aligned} \tag{18}$$

From (11), (13), (17) and (18), the expectation of RTT count in a loss cycle, i.e., $\overline{N}$, can be obtained as a function of state transition probabilities of the end-to-end path as follows

$$\overline{N} \quad = \quad \frac{5}{6} + \frac{1}{6}\left[25 + 24\left(\frac{1 - \omega}{\pi - \omega}\right)\right]^{1/2} \tag{19}$$

The duration of a loss cycle LC_i is $D_i = N_i \cdot RTT$. Therefore, its expectation is calculated as

$$\overline{D} = \overline{N} \cdot RTT \tag{20}$$

Thus, it follows from (13), (19) and (20) that the asymptotic throughput for the desired source behavior as a function of RTT and the state transition probabilities of the end-to-end path model can be expressed as

$$T = \frac{1}{4 \cdot RTT} \cdot \left(3 + \sqrt{25 + 24\left(\frac{1-\omega}{\pi - \omega}\right)}\right) \tag{21}$$

Note that the obtained throughput equation depends on state probabilities of the end-to-end path model developed in the Section 2.3. The throughput value as an output of (21) varies with RTT, π, and ω. The equation (21) gives the steady-state throughput that could be achieved if the TCP source would behave as desired. Hence, (21) is used as the driving equation for the analytical rate control operation of the ARC protocol.

2.5 ARC: The Analytical Rate Control Scheme

ARC is an end-to-end analytical rate control protocol that uses the throughput equation (21) developed in Section 2.4. The objective of ARC is to perform rate control for real-time traffic over wireless links in order to produce TCP-friendly traffic flows while maintaining high throughput performance. Its equation-based approach provides basis for multimedia traffic support in terms of time-constraint such as delay and/or jitter bounds.

ARC can run on top of RTP/RTCP [94] and UDP as shown in Figure 9 to provide real-time streaming support. ARC is not an ARQ protocol and the source does not perform retransmission due to tighter time constraints of the multimedia flows. While the rate control is mainly performed by the source, the receiver sends back an acknowledgment (ACK) for any received packet. If the RTP/RTCP is implemented, then these ACKs can be the receiver reports (RR) that includes the reception quality statistics such as the number of packets received, RTP timestamp, fraction lost, and cumulative number of packets lost. Hence, ARC source can obtain total packet loss rate, π, and round-trip time, RTT, from

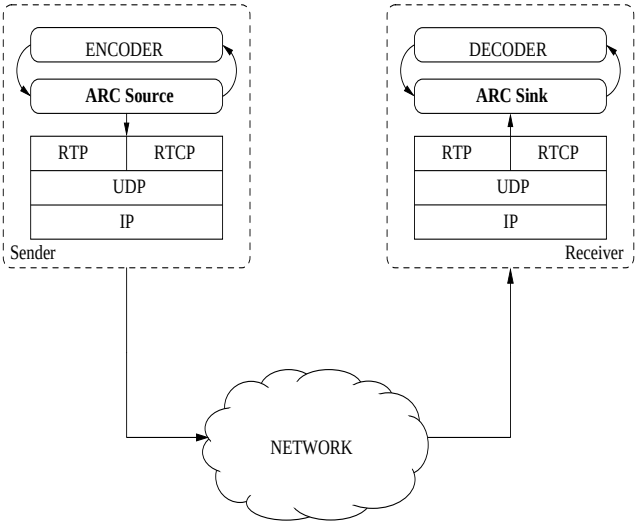

Figure 9: ARC protocol structure.

the RTP receiver reports [94]. If RTP/RTCP is not used for any reason, then data ACKs would be sufficient to acquire π and RTT.

The overall ARC protocol operation consists of two sequential periods, i.e., *Probe period* and *Steady period*. In the beginning of the connection, the information about the total packet loss rate π and the probability of packet loss due to wireless link errors ω are not available at the source. Hence, the data transmission rate, S, cannot be controlled by using (21) until the required information is obtained. During this period, the ARC source calls the `Probe()` algorithm in order to determine the initial data transmission rate. The *probe period* lasts $2 \cdot RTT$. At the end of this period, the data transmission rate S is set to initial data rate S_{Init} and the source goes into the *steady period* calling the `Steady()` algorithm. The details of the algorithm is presented in Section 2.5.1.

During the steady period, the data transmission rate S is controlled by using (21) as the required link information becomes available. At each loss event arrival, the required link information, i.e., π, ω, and RTT are obtained and the data transmission rate S is set by using (21). The details of the algorithm is presented in Section 2.5.2.

2.5.1 Probe Period

The ARC source starts a new connection by calling the `Probe()` algorithm in the probe period, which lasts $2 \cdot RTT$. In [106], low priority dummy packets are used to obtain available resource information at the beginning of the connection. Similarly, ARC source transmits only one data packet at first and then low priority *probe packets* in the first RTT with the target transmission rate, S_{Target}. Note that the target data rate S_{Target} could also be set by the application to achieve highest quality real-time streaming.

The low priority probe packets can be distinguished by one of the eight bits of the IP *type of service* (TOS) field in the IP header. While there exists considerable amount of ongoing research on the available bandwidth estimation techniques [67], ARC protocol performs link probing only in the initial phase of the connection using low priority *probe packets*. Currently, most of the commercial routers already have the *priority-queuing* capability based on the TOS field of the IP packet header [36]. While some routers in the Internet do not currently apply any priority policy, in the near future, the next generation Internet will support quality of service through the *Differentiated Service Model* (DiffServ) [25], which requires all routers to support multiple service classes with different service priorities and drop precedences. However, note that ARC does not specifically assume or require DiffServ architecture, rather a simple priority-queuing mechanism with two priority levels suffices for the *probe packet* mechanism.

The steps of the `Probe()` algorithm is shown in Figure 10. For the ARC connection starting at $t = 0$, the ARC source sends probe packets at rate S_{Target} during $0 \leq t \leq RTT$. If there is a congestion, low priority probe packets are discarded first by the priority policy deploying routers. Otherwise, they are ACKed back to the sender yielding the initial resource availability information on the path. The ARC source counts the number of ACKs, n_{Ack}, received for the transmitted probe packets during $RTT \leq t \leq 2 \cdot RTT$. Consequently, at the end of the second RTT period, i.e., $t = 2 \cdot RTT$, the ARC source sets its initial transmission rate to

$$S_{Init} = \frac{max\{1, n_{Ack}\}}{RTT} \tag{22}$$

Probe()

```
Send DATA_PACKET;
While (t ≤ RTT)
  Send PROBE_PACKET with S_Target;
end;
If (t ≥ 2 · RTT) then
  Obtain n_Ack;
  S = max{1, n_Ack}/RTT;
end;
  Steady();
end;
```

Steady()

```
Foreach (LOSS_EVENT)
 Calculate π, ω;
 Obtain RTT;
 S = 1/(4·RTT) · (3 + sqrt(25 + 24 ((1−ω)/(π−ω))));
end;
end;
```

Figure 10: The `Probe()` and `Steady()` algorithms.

2.5.2 Steady Period

The ARC source continues a connection by calling `Steady()` algorithm at $t = 2 \cdot RTT$. After the probe period, the steady-state operation of the ARC rate control depends on (21), and hence the total packet loss probability, π, the probability of packet loss due to wireless link ω, and RTT.

During this period, the ARC source uses (21) to control the data transmission rate S. To utilize (21) for this goal, the ARC source continuously obtains the packet loss rate π from RTP receiver reports (or it measures π within sliding time window of τ using ACKs, if RTP/RTCP is not available). RTT can be obtained from RTP/RTCP sublayer at the sender, or from the ACK packets. The probability of packet loss due to wireless link ω is assumed to be obtained from the lower layers, i.e., the underlying MAC layer, which already has this information, or analytically from information available at the physical layer of the mobile node [130], [34]. Note that ω includes all wireless link related packet losses encountered by the underlying MAC layer due to bit errors, fading, and signal loss due to handoff or blackout. At each loss event arrival, the required variables π, ω, and RTT are obtained and the data transmission rate S is set by using (21). For the cases

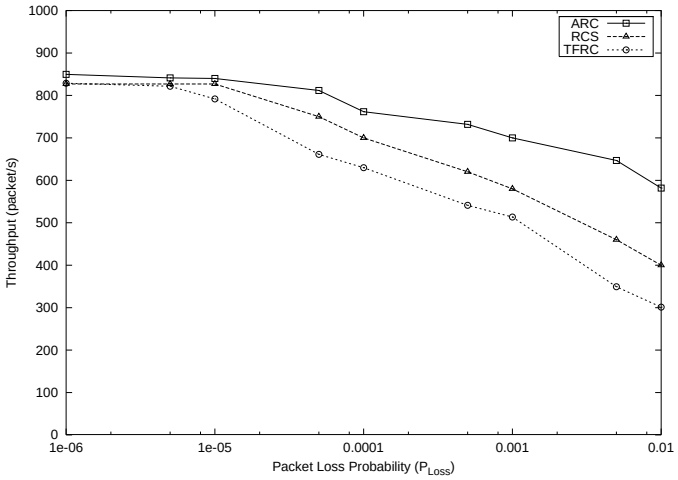

Figure 11: Throughput performance of ARC, RCS and TFRC for varying packet loss probability P_{Loss}.

where the sender is not mobile station, the information regarding to the wireless portion of the end-to-end path, i.e., the probability of packet loss due to wireless link, ω, should be informed to the sender.

The overall operation of the `Steady()` algorithm of ARC rate control scheme is shown in Figure 10. During the steady period, at each loss event (`LOSS_EVENT`) the source obtains the required link information, i.e., π, ω, and RTT. Having all the required information, the ARC source can then set its transmission rate S using the control equation, that is,

$$S = \frac{1}{4 \cdot RTT} \cdot \left(3 + \sqrt{25 + 24 \left(\frac{1-\omega}{\pi - \omega} \right)} \right) \tag{23}$$

As a result, the connection starts with the probe period and then continues with the steady period. The rate control during the steady period is performed by the rate control function (23). At each loss event, the new π, ω, and RTT are obtained and then the rate control (23) is invoked.

On the other hand, ARC rate control protocol can be in co-operation with an adaptive media encoding process. The adaptive encoder can be provided with the available bandwidth $S(t)$ estimated by the ARC as shown in Figure 9. Hence, the real-time multimedia encoder can then adapt its encoding rate $R(t)$ according to the controlled fair share of the network resources. By this way, the quality of the encoded multimedia can vary homogeneously and smoothly without leading to congestion and undesirable abrupt quality

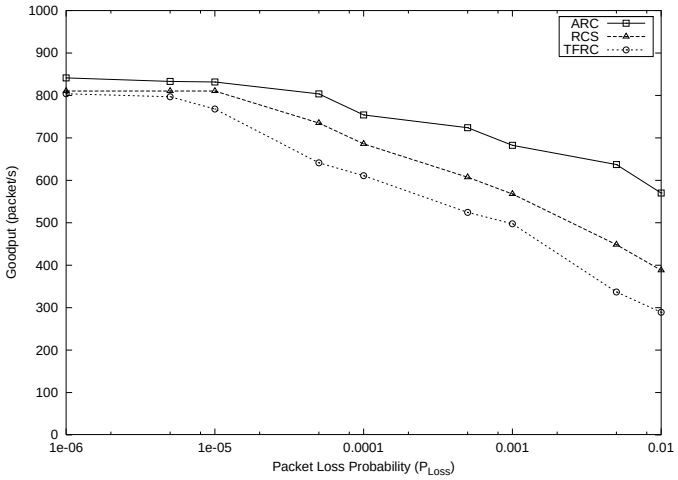

Figure 12: Goodput performance of ARC, RCS and TFRC for varying packet loss probability P_{Loss}.

variations.

2.6 Performance Evaluation

The ARC is a rate control scheme for real-time traffic over wireless links. It is intended to achieve high throughput and multimedia support for real-time traffic flows while preserving fairness to the TCP sources sharing the same wired link resources. Hence, in order to investigate the performance of the ARC, extensive simulation experiments are conducted. ARC performance is investigated in terms of the throughput, goodput, fairness, and real-time multimedia support.

2.6.1 Throughput Performance

The topology given in Figure 13 is simulated, where N ARC sources are connected to the IP backbone via wireless access points. The corresponding N receivers are assumed to be fixed terminals in the network. For the simulation purposes, $N = 10$ and the capacity of all links are set to $C = 1000$ packets/sec.

The throughput achieved by ARC, RCS (Rate Control Scheme) [106] and TFRC (TCP-Friendly Rate Control) [52] sources are observed for varying packet loss probability P_{Loss}. The preliminary version of the RCS protocol [106] is chosen for performance comparison, since it gives the best performance in lossy links. Since TFRC is also an equation-based

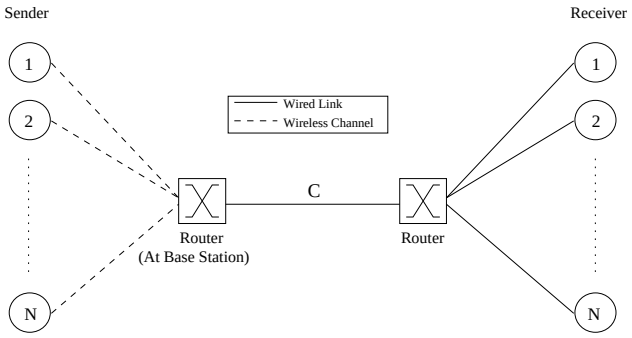

Figure 13: Simulation scenario for throughput performance evaluation.

congestion control protocol which uses TCP response function as its control equation, ARC performance is also compared to TFRC. The connections passing through wireless channel are assumed to experience packet losses due to link errors with probability P_{Loss}.

In Figure 11, the throughput of the ARC, RCS [106] and TFRC [52] protocols are shown for different P_{Loss} values. Here, for low P_{Loss} values the same throughput is achieved by all (ARC, RCS and TFRC) protocols. However, for increasing P_{Loss} values, the throughput starts to decrease as expected but with different slopes. For example, for $P_{Loss} = 10^{-2}$, ARC improves throughput over TFRC and RCS by approximately 93% and 45%, respectively.

The goodput achieved by the ARC, RCS, and TFRC are also shown for different P_{Loss} values in Figure 12. A similar pattern with throughput is observed in this scenario. For $P_{Loss} = 10^{-2}$, ARC achieves goodput higher than TFRC and RCS by approximately 92% and 44%, respectively.

Note also that the difference between the goodput and the throughput achieved by the ARC sources increases as the wireless channel conditions get worse. In the worst case, all packets transmitted by the source can get lost in the wireless link for a certain period of time. Most of the media encoding techniques, on the other hand, are predictive and hence based on the previously encoded portion of the multimedia stream. Therefore, the encoding process can continue due to real-time constraints of the multimedia source even in the presence of excessive wireless link errors. However, advanced adaptive multimedia encoders that can adjust the encoding process according to the wireless channel conditions can be used at the application layer to improve the efficiency of the real-time multimedia

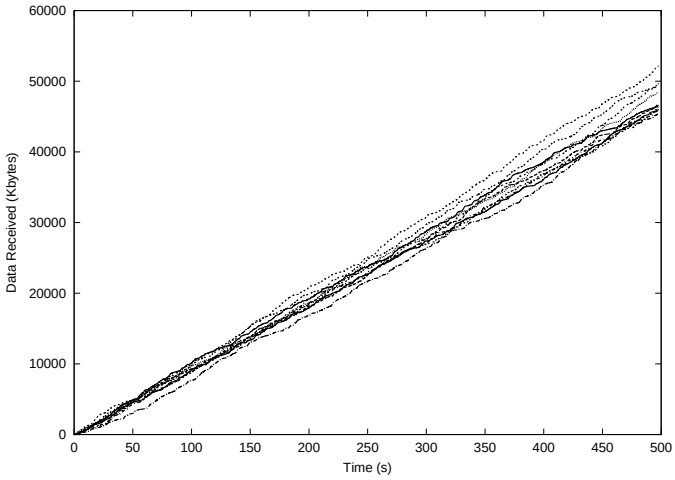

Figure 14: Fairness among the ARC protocol sources: Data received at each ARC sink with respect to time.

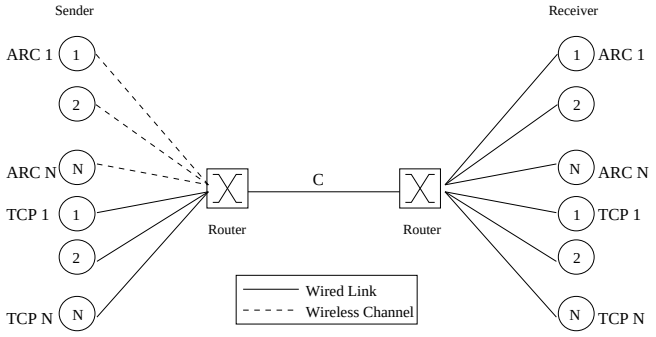

Figure 15: Simulation scenario for heterogeneous fairness evaluation.

encoding and hence transmission over the wireless links [119]. However, recall that ARC is a transport layer rate control protocol, which adjusts the data transmission rate such that the real-time multimedia flows use fair share of the link resources. Hence, the details of the efficient adaptive multimedia encoding based on the wireless channel state are left beyond the scope of this work.

2.6.2 Fairness

For the fairness performance of ARC protocol, two different scenarios, i.e., homogeneous and heterogeneous fairness, are considered. For the homogeneous case, the fairness among ARC flows in sharing bottleneck is observed. For heterogeneous case, the fairness of ARC protocol is explored, where ARC sources share the bottleneck link with wired TCP sources.

1. **Homogeneous scenario:** In this case, the same topology shown in Figure 13 is simulated with $N = 10$ and link capacity $C = 1000$ packets/sec. Here, all of the senders deploy the ARC rate control scheme and share the same bottleneck. In Figure 14, the data received at each ARC sink is shown as a function of time t. It is observed from Figure 14 that all sinks receive almost the same amount of data at any point in time during the simulation period. Therefore, it is concluded that the ARC sources are all given a fair share of the network resources.

2. **Heterogeneous scenario:** In this case, the topology shown in Figure 15 is simulated, where $N = 10$ ARC connections and $N = 10$ TCP-NewReno [51] connections share the same bottleneck with capacity $C = 1000$ packets/sec. Here, ARC sources are connected to the bottleneck via wireless channel (dashed lines in Figure 15) experiencing $P_{Loss} = 10^{-3}$ due to wireless link errors. TCP sources are connected to the bottleneck via wired links (solid lines in Figure 15), hence they do not experience any wireless link errors. The reason for simulating such topology is basically to observe the fairness performance of ARC to the wired TCP sources sharing the same bottleneck link.

   Let Φ be the fairness between the ARC and TCP sources, where Φ is the ratio of the average throughput of ARC (T_{ARC}) and TCP (T_{TCP}) connections, i.e.,

$$\Phi(t) = \frac{T_{ARC}(t)}{T_{TCP}(t)} \tag{24}$$

   Therefore, if $\Phi(t) \approx 1$, then ARC and TCP sources are given a fair share of bottleneck. In Figure 16(a) and 16(b), the average data received by the ARC and TCP receivers and their ratio $\Phi(t)$ as a function of time are shown, respectively. As shown in Figure 16(a), the wired TCP sources are always given a higher share of the bottleneck than the ARC sources. In Figure 16(b), $\Phi(t) < 1$ for all $t < 0$ which concludes that ARC preserves fairness to the wired TCP sources sharing the same wired link resources while significantly enhancing network utilization efficiency.

   The simulation experiments are also performed using the same simulation environment shown in Figure 15, where, in this case, TCP sources also traverse wireless links

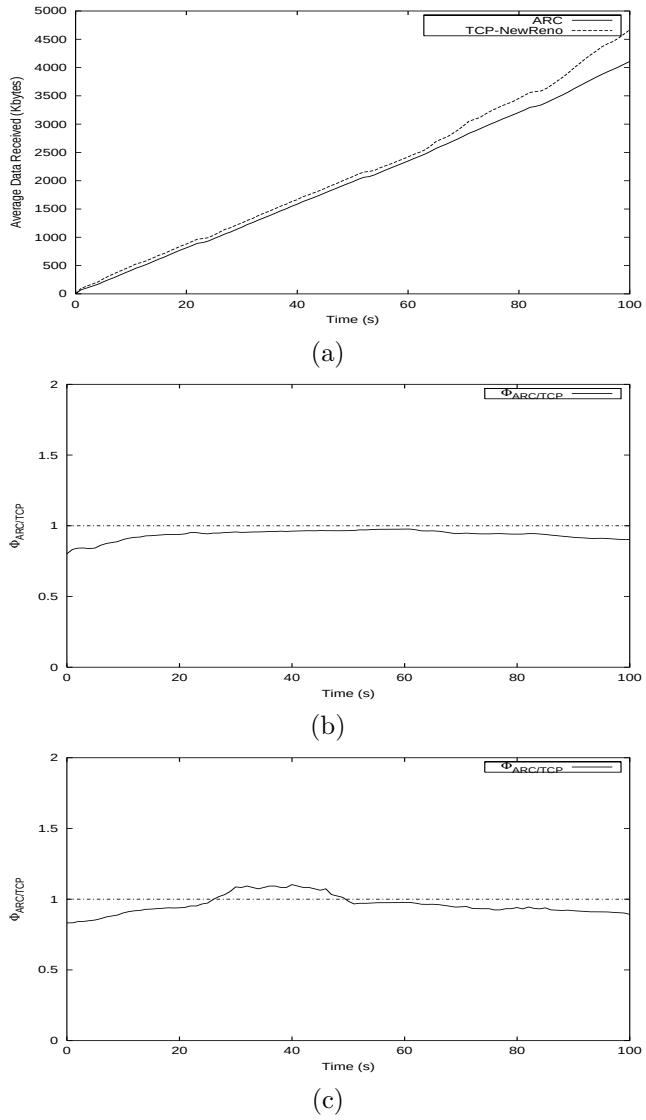

Figure 16: Fairness to wired TCP sources: (a) Average data received at each ARC and TCP-NewReno sink (b) ratio of the received data with respect to time. Fairness to wireless TCP sources: (c) ratio of the received data with respect to time for the cases where TCP sources also traverse wireless links.

as ARC sources. The objective of this set of simulation experiments is to observe how ARC sources maintain fairness after temporary wireless link errors are over, and consequently, how TCP sources recover from unnecessary rate decrease and start to obtain the fair share of the link resources.

Here, wireless link errors are introduced with $P_{Loss} = 10^{-3}$ at $t = 25$ s for a duration of 20 seconds. As it is observed in Figure 16(c), TCP sources have higher share of the

bottleneck until the wireless link errors are introduced. After $t = 25$ s, ARC sources start to obtain higher link resources. During 25 s $< t <$ 45 s, ARC sources utilize higher bandwidth than TCP sources. This is because TCP sources unnecessarily perform rate decrease due to misinterpretation of packet losses due to wireless link errors as congestion losses as discussed in Section 2.4. Note that effective loss-labeling schemes such as in [21] can improve the TCP performance in wireless networks by enabling TCP sources to more accurately identify packet loss reasons. By this way, the transient unfairness of ARC sources to TCP sources observed in Figure 16(c) due to the inherent shortcomings of TCP can be further minimized.

However, as shown in Figure 16(c), as the wireless link errors are over, the link share of the ARC sources decreases. This is because TCP sources start to recover from the erroneous rate decrease they performed in case of wireless link errors. As shown in Figure 16(c), after $t = 50$ seconds, $\Phi < 1$. Therefore, ARC sources can maintain the fairness after the wireless link errors are over and hence TCP sources can recover their fair share while competing with ARC sources for the same bottleneck resources.

2.6.3 Multimedia Support

In order to investigate ARC performance in terms of multimedia time-constraints, the rate change pattern and delay variation, i.e., jitter, are observed One of the most important properties of equation-based rate control is its smoothly changing data transmission rate. The transmission rates of ARC, RCS (Rate Control Scheme) [106], TFRC (TCP-Friendly Rate Control) [52] and TCP-NewReno [51] sources are shown in Figure 18 as a function of time.

In this scenario, simulation experiments with ARC, RCS, TFRC and TCP-NewReno sources are performed using the same simulation scenario given in Figure 13. In the experiments, the sources are in wireless domain experiencing packet loss probability $P_{Loss} = 10^{-2}$. As shown in Figure 18, the most frequent rate change is experienced by the mobile TCP-NewReno Source. This is because the mobile TCP source interprets every packet loss as a congestion indication, hence it performs unnecessary rate decrease. TFRC also provides

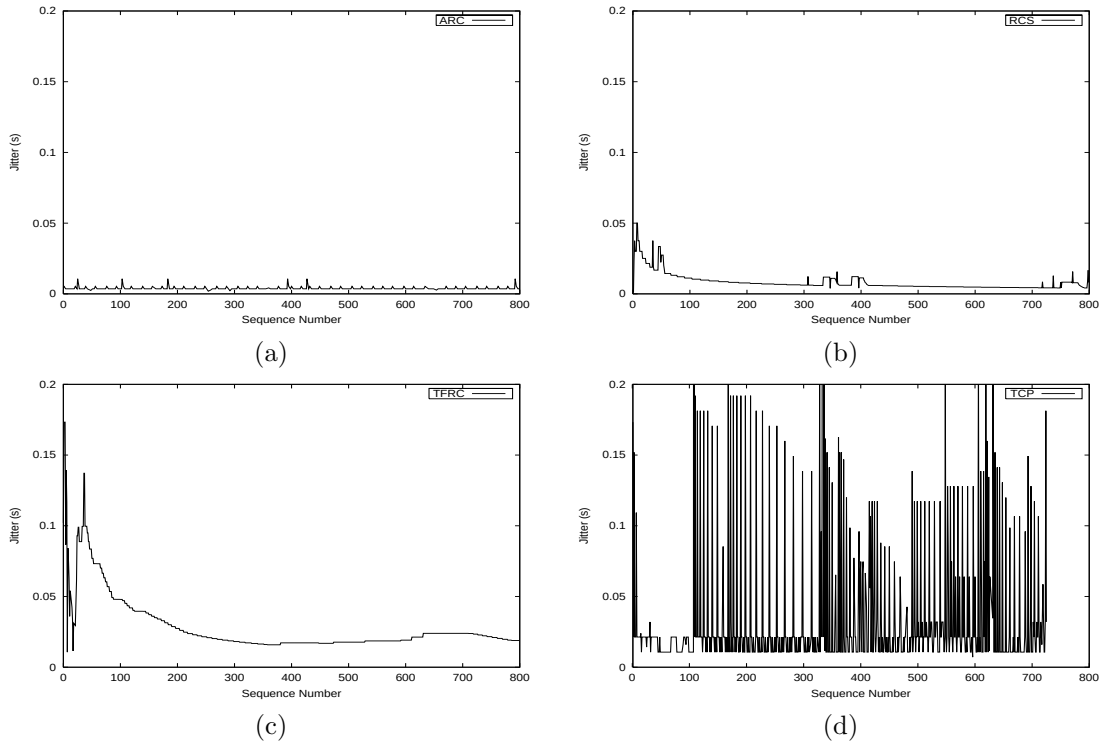

Figure 17: Jitter experienced by each packet sent by (a) ARC, (b) RCS, (c) TFRC, and (d) TCP-NewReno sources.

very smooth rate change, however, it also cannot distinguish wireless link error losses than congestion losses. Therefore, TFRC also experiences unnecessary rate decrease during connection time. Since RCS follows the conservative TCP rule, i.e., rate halving at packet loss, and then tries to recover from this rate change by using dummy packets [106], RCS source also experiences more frequent rate change compared to ARC source as shown in Figure 18.

The jitter experienced by ARC, RCS [106], TFRC [52] and TCP-NewReno [51] flows are also investigated. The jitter experienced by each packet transmitted by the ARC, TFRC and TCP sources are shown in Figure 17(a), 17(b), 17(c), and 17(d), respectively. In Figure 17(d), it is observed that the packets sent by the TCP source always experience much higher jitter than the packets sent by TFRC, RCS, and ARC. This is because TCP performs rate throttle at each packet loss hence resulting in high delay variation. RCS [106] also performs rate halving at each packet loss; however, since it recovers from unnecessary rate halving by using dummy packets, RCS source experiences less jitter than TCP source. TFRC also

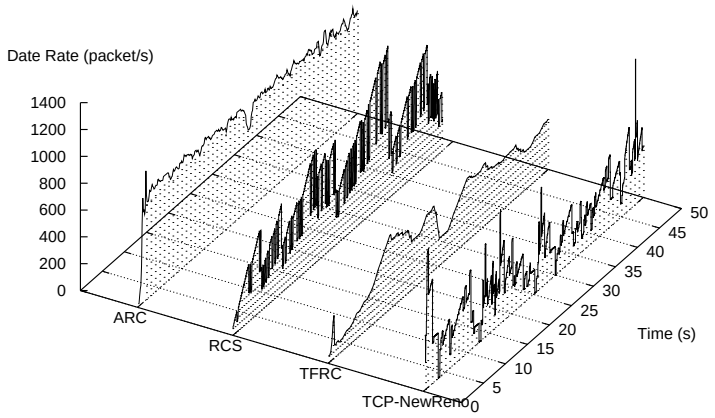

Figure 18: Transmission rate variation of ARC, RCS, TFRC, and mobile TCP-NewReno sources with time.

suffers from the wireless link errors hence makes unnecessary rate changes. As observed in Figure 17(a), 17(b), 17(c), the jitter experienced by ARC, RCS and TFRC packets is mostly around 5, 10 and 25 ms, respectively. Thus, ARC outperforms RCS, TFRC and TCP in terms of jitter-bounds of the real-time multimedia transmission over wireless links.

CHAPTER III

RATE CONTROL SCHEME FOR IP NETWORKS WITH LOSSY LINKS AND LONG ROUND TRIP TIMES

In this chapter, a new Rate Control Scheme (RCS) is introduced for real-time interactive applications in networks with high bandwidth-delay products and high bit error rates. The RCS protocol was first introduced in [106] and then revised and significantly enhanced in [4] to address the temporal signal loss conditions and real-time considerations of the adaptive multimedia applications. In Section 3.2, the current related work in the literature is extensively explored. In Section 3.3, RCS protocol algorithms are introduced. In Section 3.4, RCS operation details and behavior under different network conditions are explored: first, when a packet loss occurs due to link errors; second, when a packet loss occurs due to network congestion; third, when a packet loss occurs due to temporary signal loss. Simulation experiment results are presented in Section 3.5.

3.1 Motivation

As discussed in Section 2.1, real-time applications have strict requirements on end-to-end delay. In a shared network, such as the Internet, all traffic flows are expected to be *good network citizens* or *TCP friendly* [88], i.e.,

- *Rule 1*: Their transmission rate can increase as long as the network is not congested, and
- *Rule 2*: Their transmission rate must decrease immediately when the network is congested.

Next generation IP-routers will penalize traffic flows not compliant with these rules [48].

In case of real-time streams, transmission rate, S, can be adjusted by adapting the quality of transmitted stream based on the available bandwidth if, for example, layered

encoding is used and the source transmits the maximum number of layers that can fit in S [88]; or the encoding parameters are changed so that the output traffic rate is not higher than S. Consequently, the quality of transmitted streams adapts to network condition.

In recent years, much research has been conducted to develop rate control protocols for real-time multimedia applications in the terrestrial wireline networks [18], [26], [32], [52], [63], [69], [87], [88], [100]. The common objective of these proposed solutions is to enable the real-time multimedia applications to respond to the network congestion by controlling their bandwidth share such that the link resources are fairly shared. However, the major drawback of these solutions is that they are developed for wired links which are assumed to have negligible errors and hence these solutions cannot be directly applied to the links with high bit error rates and high bandwidth-delay products.

Packet losses are the only congestion sign in the current Internet. Accordingly, all the existing rate control schemes proposed for wireline networks decrease their transmission rates when a packet loss is detected. However, some links, such as wireless and satellite links, are characterized by high link error rates and thus, packet loss can occur due to link errors with probability even higher than 10^{-2} [20]. If the source decreases its transmission rate when a packet loss occurs due to link errors, then the network efficiency decreases drastically [72, 85], i.e., it can be lower than 20%. This problem is amplified by the long delays involved in most Internet communications. For example, Table 1 shows the round trip time, RTT, values measured when the mail server of the University of Catania (Italy) is connected to the WEB servers of other universities. Moreover, even higher RTT values have been observed in Wireless Wide Area Networks (WWAN). A typical round-trip time (RTT) varies between few hundred milliseconds and seconds in cellular WWAN links due to the extensive physical layer processing, e.g., forward error correction (FEC), interleaving and transmission delays [60]. The access delay is much higher in satellite links, which have high propagation delay up to 270 ms [10].

Delay can also be high because of the high hop distance between the two end systems. In fact, each hop causes a new queuing and processing delay. In Figure 19 the current hop distance probability distribution is shown [79]. The current average hop distance is about

Table 1: *RTT* values for long distance connections.

University	Country	*RTT*
Georgia Tech	USA	200 msec
University of Campinas	Brasil	420 msec
Korea University	Korea	430 msec
Beijing University	China	800 msec

16 and is expected to increase in the future [79].

Consequently, despite the existence of many proposed rate control schemes for real-time multimedia flows in the terrestrial wireline networks, there still exists a need for a new rate control scheme that can address the adverse effects of the high bit error rates and high bandwidth-delay products. In order to address this need, in this chapter, a Rate Control Scheme (RCS) is proposed for real-time traffic in networks with high bandwidth-delay products and high bit error rates.

RCS is an end-to-end rate control scheme which produces TCP-friendly traffic flows for real-time interactive applications as well as real-time multimedia streaming and improves throughput performance in networks with high bandwidth-delay products and high bit error rates. The entire RCS protocol operation consists of four main algorithms. The connection starts by calling the *Initial* algorithm which is specifically tailored to capture the link resources in a fast and controlled manner in the links with high bandwidth-delay products. This is achieved by probing the availability of the network resources in the connection path by using low priority *dummy packets*. In fact, RCS is based on the idea of using low priority packets called *dummy packets* [10]. This approach requires all the routers in the connection path to apply some priority policy. If a router on the connection path is congested, then it discards low priority dummy packets first. If the routers are not congested, then dummy packets can reach the destination which then sends ACKs back. When the source receives an ACK for a dummy packet, this is the evidence that there are still unused resources in the network. Hence, RCS source exploits this information to set its transmission rate in the Initial State accordingly. After the initial data transmission rate is set by the Initial State algorithm, RCS source executes *Steady* algorithm which performs Additive-Increase

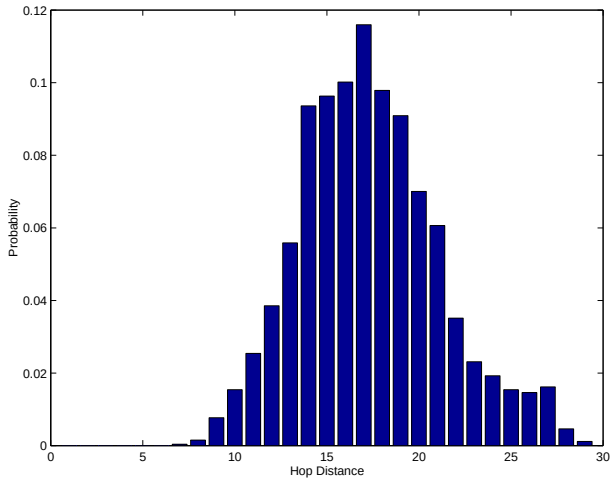

Figure 19: Hop Distance Distribution.

Multiplicative-Decrease (AIMD) rate control. In case of a packet loss, RCS source calls *Detected* algorithm which is developed to differentiate packet losses due to congestion and link errors by accurately capturing the actual link resources using dummy packets. In this algorithm, the data rate is first halved following conservative TCP behavior for TCP-friendliness purposes. Based on the ACKs received for the transmitted dummy packets, the source can distinguish congestion and link error related packet losses and hence recover from the rate throttle in case of link errors. Furthermore, in order to address the possible temporal signal loss conditions, RCS incorporates *Backoff* algorithm into its protocol operation. Simulation experiments show that RCS achieves high throughput performance without penalizing TCP connections in networks with high bandwidth-delay products and high bit error rates.

3.2 Related Work

The need of rate control for multimedia flows in the Internet has been subject of much research in the past. Three approaches for rate control of real-time traffic in the Internet can be distinguished:

1. *TCP-like protocols*: Rate control is performed as in TCP. Packet losses are interpreted as the clue for network congestion and the transmission rate is halved accordingly.

2. *Equation based protocols*: Statistics on the round trip time and packet loss probability are utilized to estimate the available bandwidth in the network. The source sets its

transmission rate accordingly.

3. *Explicit bandwidth notification-based protocols*: The network elements evaluate the available bandwidth which is notified to the traffic source. The transmission rate is set accordingly.

TCP-like protocols [18], [32], [63], [87], [88], [100] follow AIMD data rate control. In [63], a congestion control scheme, which follows TCP behavior without retransmissions, is proposed for real-time multimedia streams in the terrestrial Internet. The *Streaming Control Protocol* (SCP) [32] is a modified version of TCP [64] that performs TCP-Vegas-like rate adjustment. LDA [100] performs flow control for real-time streams by using mechanisms very similar to those of TCP [64]. RAP [88] is an AIMD rate control protocol proposed for unicast playback real-time streams. It additively increases the data rate periodically as long as there is no congestion, and halves its data rate when a packet loss is detected with its ACK-based loss detection mechanism. All of these solutions have been developed for rate control in terrestrial wired networks where link errors are negligible. Therefore, in case of packet losses due to link errors they follow the conservative rate halving behavior of TCP, which assumes reliable and almost error-free communication link presence. This is unnecessary and causes severe performance degradation. Furthermore, the recovery from such unnecessary transmission rate decreases takes a time period proportional to the RTT. Hence, the above problem is amplified in case of long propagation delays.

Considerable amount of research has been devoted to equation-based congestion control in recent years [26], [52], [125]. Instead of responding to a single packet loss, equation-based congestion control uses a control equation to adjust the transmission rate. As an example, TCP-Friendly Rate Control (TFRC) [52] is one of the most important equation-based congestion control proposals in the literature. The control equation of TFRC is the TCP response function [83] governing the steady-state transmission rate of TCP as a function of round-trip time, RTT, and loss event rate, p. However, the existing equation-based rate control schemes cannot be applied to the links with high bit error and high propagation delays. The fundamental reason is that the throughput equation in [83] models

the steady-state TCP behavior over error-free wireline links. This approach is again based on the assumption that packet loss is an indication of congestion and hence is directly related to the actual link resources. Therefore, it may result in inaccurate rate estimation and hence underutilization of the link resources. Furthermore, the TCP throughput equation in [83] is inversely proportional to the RTT. As a result, the performance of equation-based rate control schemes severely degrades when links with high propagation delay are involved in the end-to-end communication.

Finally, schemes have been proposed which are based on explicit notifications by the network elements. As an example, in [69] an adaptive congestion control scheme for real-time video transport is presented where the video source periodically receives explicit feedback from the network about the amount of buffer and the service rate the source can obtain. Based on this information, the video source controls the transmission rate by adjusting the quantization factor (Q) of the video encoder. These schemes require each router to continuously perform per flow buffer and bandwidth monitoring and to send explicit feedback. Furthermore, this method also does not consider link errors which may avoid the arrival of the required explicit feedback and lead to erroneous transmission rate estimation.

3.3 RCS: Rate Control Scheme

3.3.1 Protocol Overview

RCS is an end-to-end rate control scheme which uses AIMD [35] congestion control approach, in order to produce TCP-friendly traffic flows while maintaining high throughput performance in networks with high bandwidth-delay products and high bit error rates.

RCS runs on top of RTP/RTCP [94] and UDP and is mainly implemented at the source but also needs some functions at the destination. In fact, at the destination RCS layer sends back an acknowledgement (ACK) for any received packet, as suggested also in [88]. Note that these ACKs are used only for flow control as will be explained in the following. No retransmissions are performed. At the destination, RCS layer passes the received data packet to the decoder.

RCS protocol consists of four main algorithms specifically tailored to address the real-time multimedia transport in the networks with high bit error rates and high bandwidth-delay products.

The time needed to reach a certain data transmission rate in the beginning of the connection is proportional to the RTT of the end-to-end path. For this reason, it is well known that link resources in the connection path may not be utilized for a long period in the links with high propagation delay such as satellite links [10]. In order to effectively handle the adverse effects of the high propagation delay, RCS contains *Initial* algorithm which is specifically tailored to capture the link resources in a fast and controlled manner in the links with high bandwidth-delay products. The details of the Initial State behavior are explained in Section 3.3.3.

The link probing technique exploited in the Initial State is based on the usage of low priority *dummy packets* [10]. This approach requires all the routers in the connection path to apply some priority policy. If a router on the connection path is congested, then it discards low priority dummy packets first. If the routers are not congested, then dummy packets can reach the destination which then sends ACKs back revealing the existence of unused resources in the network. RCS uses low priority dummy packets in order to minimize the effect of probing traffic on the throughput of the actual data flows. The low priority *dummy packets* concept is explained in detail in Section 3.3.2

After the initial data transmission rate is set by the Initial State procedure, RCS source executes *Steady* algorithm which performs additive-increase rate control. In order to address the challenges due to high bit error rates, RCS source calls *Detected* algorithm in case of a packet loss. Since the cause of the packet loss is unknown to the source at the beginning, RCS decreases the data rate to preserve TCP-friendly behavior. Hence, in the Detected State, RCS source halves the data rate and starts transmitting dummy packets. Depending on the ACKs received for the dummy packets, the source can recover from the rate throttle in case of link errors. The details of the *Steady* and *Detected* algorithm operations are presented in Sections 3.3.4 and 3.3.5, respectively.

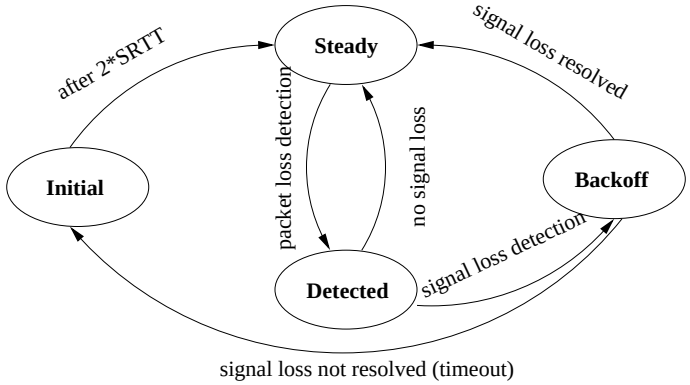

Figure 20: Finite State Machine Model of the RCS Source.

Furthermore, in order to address the possible temporal signal loss conditions, RCS incorporates *Backoff* algorithm into its protocol operation. The objective of the Backoff algorithm is to capture the link loss condition and hence minimize the performance degradation by avoiding unnecessary rate throttles. The Backoff algorithm operation is discussed in Section 3.3.6.

3.3.2 Dummy Packets

RCS is based on the use of dummy packets. Dummy packets are low priority packets used by the source to probe the availability of network resources [10]. If a router on the connection path is congested, then it discards the IP packets carrying dummy packets first. Consequently, the transmission of dummy packets does not cause a decrease in the throughput of actual *data packets*. If the routers are not congested, then the dummy packets can reach the destination which sends the related acknowledgments. The ACKs for dummy packets are also carried by the low priority IP packets. The source interprets the ACKs for dummy packets as the evidence that there are unused resources in the network and accordingly, can increase its transmission rate. Note that dummy packets may produce some overhead, but it is demonstrated in Section 3.5 that they use resources which otherwise would be unutilized.

The new scheme requires all routers in the connection path support some priority discipline. In fact, RCS injects dummy packets into the network regardless of the current traffic load. As a consequence, dummy packets may congest routers and affect data packet

throughput if a router on the connection path does not apply any priority policy. Note that in traditional IP [86] networks the IP *type of service* (TOS) can be used for this purpose. In fact, one of the eight bits of the TOS field in the IP header gives the priority level of the IP packet [86]. Instead, more recent IP versions, e.g., IPv6 [41], explicitly provide several priority levels. Therefore, since both dummy packets and their ACKs are encapsulated by low priority IP packets, RCS source and the receiver can distinguish both dummy packets and the ACKs for dummy packets from the normal data packets via TOS field in the IP packet header.

Currently, most of the commercial routers, e.g., Cisco series 2500, 4500 and higher [36], already implement *priority-queuing* capability and use the TOS field of the IP packet header.

Moreover, the trend for the Internet is to evolve towards a QoS-capable network by means of the *Differentiated Service Model* (DiffServ) [25], which requires all routers to support multiple service classes with different service priorities and drop precedences. The details of the various priority-queuing methods are beyond the scope of the work.

Applications generating low priority traffic may be penalized by dummy packets even if priority is supported by routers. The transmission of dummy packets may cause congestions for low priority traffic. However, dummy packets are not continuously transmitted rather they are transmitted for short periods of time. Hence, those possible congestion situations last for a short period. In fact, dummy packets are transmitted only in two cases:

- *In the beginning of a new connection.* This occurs only once for each connection and the RCS source transmits dummy packets for a period of one round-trip time approximately.

- *When a data packet loss is detected.* Packet losses can be due to network congestion, link errors or temporal signal loss conditions. For the first two cases RCS transmits dummy packets for a period no longer than one round trip time. In the temporal signal loss conditions, RCS sends dummy packets until the signal is detected again. Moreover,

  - If the packet loss was due to network congestion, then high priority traffic is

already using all the resources, i.e., no low priority traffic is passing through the connection.

– If the packet loss was due to link errors or temporal signal loss, the transmitted dummy packets may harm low priority data traffic. However, it will be shown that the payback for this problem is a high increase of the throughput of high priority traffic.

On the other hand, the traffic generated by the data and the low-priority dummy packet transmissions are assumed not to be isolated in the network. Therefore, the data packets and low-priority dummy packets do not experience difference service rates rather they experience different packet loss rates due to different drop preferences. In fact, the simulation experiment results show that RCS achieves high throughput performance for a wide range of varying background traffic scenarios as will be presented in Section 3.5.2.

As shown in Figure 20, RCS source is a finite state machine model with four states: *Initial*, *Steady*, *Detected* and *Backoff*. In the following the behavior of RCS source is presented in each of the above states.

3.3.3 Initial State Operation

In the beginning of a new connection, the source must set the initial transmission rate value, $S_{Initial}$. Let $S_{Available}$ be the transmission rate sustainable by the network. The choice of $S_{Initial}$ is important, because

- If $S_{Initial} >> S_{Available}$, then the new connection will cause network congestion.
- If $S_{Initial} << S_{Available}$, then resource utilization is low and will stay low for a time interval proportional to the bandwidth-delay product.

In order to effectively handle the adverse effects of the high propagation delay and hence efficiently set the transmission rate value S, RCS incorporates the `Initial()` algorithm given in Figure 21 which is the new procedure specifically tailored to capture the link resources in a fast and controlled manner in links with high bandwidth-delay products.

RCS starts a new connection in the Initial state and remains there for a time interval, $t_{Initial}$, equal to two times the estimated round trip time, $SRTT$, i.e., ($t_{Initial} = 2 \cdot SRTT$). Hence, the objective of Initial state phase is to set the initial transmission rate as early as $t = 2 \cdot RTT$ without leading to any congestion. RCS maintains an $SRTT$ value as in case of TCP [64]. There are different methods to compute $SRTT$, e.g., [64, 70]. However, note that $SRTT$ is only utilized as a reference timescale here, thus, RCS performance is independent of the $SRTT$ selection.

During the Initial state phase, RCS source executes the `Initial()` algorithm shown in Figure 21.

```
Initial()
    t_0 = t;
    t_1 = t_0 + SRTT;
    t_END = t + 2 · SRTT;
    IPG_Dummy = 1/S_Target;
    t_next_dummy = t_0 + IPG_Dummy;
    n_ACK = 0;
    while (t ≤ t_END)
        while (t ≤ t_1)
          while(t < t_next_dummy)
             if (DUMMY_ACK_ARRIVAL)
                n_ACK = n_ACK + 1;
             end;
          end;
          send(DUMMY_PACKET);
          t_next_dummy = t_next_dummy + IPG_Dummy;
       end;
       if (DUMMY_ACK_ARRIVAL)
          n_ACK = n_ACK + 1;
       end;
    end;
    wdsn = −1;
    S = max(1, n_ACK)/SRTT;
    state=Steady;
end.
```

Figure 21: `Initial()` Algorithm.

Let t represent the current system time and t_0 be the initial time instant. The Initial phase lasts for two times $SRTT$, as a result, the `Initial()` algorithm will terminate at the time t_{END}, given by:

$$t_{END} = t_0 + 2 \cdot SRTT \tag{25}$$

Let S_{Target} represent the value of the data transmission rate needed to transmit the real-time stream with the highest quality. For example,

- If layered encoding is used, then S_{Target} is the transmission rate needed to transmit all the layers of the encoded stream.
- If adaptive encoding is used, then S_{Target} is the transmission rate needed to transmit the real-time stream encoded with the highest definition.

During $t_0 \leq t \leq (t_1 = t_0 + SRTT)$, RCS source sends dummy packets (`send(DUMMY_PACKET)`) at rate S_{Target}, i.e., the inter-transmission time is ($IPG_{Dummy} = 1/S_{Target}$). RCS uses the variable t_{next_dummy} to indicate the time when the next dummy packet has to be sent. After each dummy packet transmission, t_{next_dummy} is updated as shown in Figure 21.

If there are sufficient bandwidth resources available at the routers along the path, the dummy packets transmitted in $t < RTT$ are received and ACKed back by the receiver as explained in Section 3.3.2. The source interprets the ACKs for dummy packets as the evidence that there are unused resources in the network. Therefore, RCS source counts the number, n_{ACK}, of ACKs received for dummy packets. In fact, whenever it receives an ACK for a dummy packet (`DUMMY_ACK_ARRIVAL`), it increases n_{ACK} by one as shown in Figure 21.

Therefore, the number of ACKs received for the dummy packets, n_{ACK}, gives the estimate of the available network resources, in fact

$$\frac{n_{ACK}}{SRTT} \approx \min\{S_{Target}, S_{Available}\} \tag{26}$$

Consequently, at the end of the Initial state, i.e., $t = 2 \cdot RTT$, RCS source sets the data transmission rate S by using the network resource availability information captured by the dummy packets, i.e.,

$$S = \frac{\max\{1, n_{ACK}\}}{SRTT} \tag{27}$$

Furthermore, before leaving the Initial State, RCS sets the variable $wdsn = -1$. As it is clarified in the Section 3.3.5, RCS uses the variable $wdsn$ in order to preserve TCP-friendliness.

3.3.4 Steady State Operation

In the Steady state, RCS source assumes that the network is not congested. Thus, according to the additive-increase scheme [35], it increases its transmission rate in a step-like fashion periodically. Moreover, upon receiving an ACK for a dummy packet, the RCS source checks the value of the variable $wdsn$. The variable $wdsn$ is used in order to match RCS source behavior with TCP behavior when the network is congested [10]. If $wdsn$ is higher than zero, then RCS source decreases $wdsn$ by one, i.e., ($wdsn = wdsn - 1$). Otherwise, RCS source increases its transmission rate by one packet per estimated round trip time, $SRTT$. The details of the determination of the variable $wdsn$ and how it assures TCP-friendliness are explained along with the Detected state behavior in Section 3.3.5. RCS source leaves the Steady state for the Detected state when a data packet loss is detected. Note that RCS source uses the same mechanism of TCP Reno [66] to detect data packet losses.

```
Steady()
    END=0;
    t_0 = t;
    t_next_data = t_0;
    t_next_increase = t_0 + SRTT;
    while (END == 0)
       if (PACKET_LOSS_DETECTION)
          END=1;
       end;
       if (t ≥ t_next_data)
          send(DATA_PACKET);
          t_next_data = t_next_data + IPG;
       end;
       if (t ≥ t_next_increase)
          S = min(S + 1/SRTT, S_Target);
          IPG = 1/S;
       if (DUMMY_ACK_ARRIVAL)
          if (wdsn == 0)
             S = min(S + 1/SRTT, S_Target);
             IPG = 1/S;
          else
             wdsn = wdsn - 1;
          end;
       end;
    end;
    state=Detected;
end.
```

Figure 22: `Steady()` Algorithm.

During the Steady phase, RCS source executes the `Steady()` algorithm shown in Figure 22. The algorithm uses the following variables:

- `END`: It is a Boolean variable which is set to 1 when a data packet loss is detected (`PACKET_LOSS_DETECTION`) to indicate that the algorithm must be terminated and RCS source must go to the Detected State, i.e., `state`=Detected.

- t_0: It gives the time instant when the current Steady phase started.

- t_{next_data}: It is the time instant when the next data packet has to be sent. When the current time, t, is higher than or equal to t_{next_data}, a data packet is sent (`send(DATA_PACKET)`), and t_{next_data} is updated.

- IPG: It is the time interval between two successive data packet transmissions and is given by $IPG = 1/S$, where S is data transmission rate.

- $t_{next\_increase}$: It is the time instant when the transmission rate, S, must be increased. According to the additive increase scheme [35], S is increased periodically in a step-like fashion. In order to match the behavior of RCS source with the behavior of TCP [64], the period is $SRTT$ and the step height is $1/SRTT$. However, the transmission rate, S, never exceeds the value S_{Target}. As a consequence, if $t \geq t_{next\_increase}$, the transmission rate is updated as follows

$$S = \min\{S + 1/SRTT, S_{Target}\} \tag{28}$$

  and $t_{next\_increase}$ is updated

$$t_{next\_increase} = t_{next\_increase} + SRTT \tag{29}$$

- $wdsn$: When the ACK for a dummy segment is received, i.e., `DUMMY_ACK_ARRIVAL`, RCS source checks the value of $wdsn$. If $wdsn = 0$, then the transmission rate is increased as in (28), otherwise $wdsn$ is decreased by one, i.e., $wdsn = wdsn - 1$.

3.3.5 Detected State Operation

In order to address the challenges due to high bit error rates, RCS source incorporates the *Detected* algorithm. RCS source enters the Detected state when a data packet loss is detected.

In the Detected state, RCS source executes the `Detected()` algorithm shown in Figure 23. Packet losses are the only indication of network congestion in the current Internet. Since the reason for the packet loss is unknown to the source at the beginning, it is safer to decrease the data rate to preserve TCP-friendly behavior. As a result, RCS source maintains the TCP [64] conservative assumption that all packet losses are due to network congestion and, accordingly, halves its data transmission rate, S upon entering Detected state. The Detected phase lasts for a time interval equal to the estimated round trip time, $SRTT$, thus, it will finish at time ($t_{END} = t_0 + SRTT$), where t_0 is the time instant when the Detected algorithm is initiated. At the end of the Detected phase, RCS source goes back to the Steady or Backoff state as shown in Figure 20. Moreover,

- The transmission rate, S, for data packets is halved, i.e., ($S = S/2$), and IPG is updated accordingly, i.e., ($IPG = 1/S$).
- The variable $wdsn$ is set to the value $wdsn = (SRTT \cdot S)$.
- The time, t_{next_data}, of the next data packet transmission is set.
- The Boolean variable $ack\_received$ is initially set to false $ack\_received = false$. If any ACK is received during the Detected state (`ACK_ARRIVAL`), this variable is set to true, i.e., $ack\_received = true$.

Furthermore, the sender also transmits dummy packets in order to differentiate packet losses due to congestion and link errors by probing the actual availability of network resources. In the Detected phase, data packets are sent with rate S and two dummy packets are transmitted for each data packet. Packet transmissions are uniformly distributed, thus, the time interval between two successive transmissions is $IPG/3$. The reason for transmitting two dummy packets for each data packet is explained below.

```
Detected()
    t_0 = t;
    t_END = t_0 + SRTT;
    S = S/2;
    IPG = 1/S;
    t_next_data = t_0;
    wdsn = SRTT · S;
    ack_received = false;
    while (t ≤ t_END)
       if (t ≥ t_next_data)
          send(DATA_PACKET);
       if (t ≥ t_next_data + IPG/3)
          send(DUMMY_PACKET);
       if (t ≥ t_next_data + 2 · IPG/3)
          send(DUMMY_PACKET);
       t_next_data = t_next_data + IPG;
       if (ACK_ARRIVAL)
          ack_received=true;
    end;
    if (ack_received ==true)
       state=Steady;
    else
       state=Backoff;
end.
```

Figure 23: `Detected()` Algorithm.

The ACKs for the dummy packets transmitted in the Detected state are received in the Steady state. Let S_0 be the data transmission rate when the packet loss occurs. If the packet loss is due to congestion, then the congested router can serve $S_0 \cdot SRTT$ packets per round-trip time, approximately. As a result, the network will accommodate $S_0 \cdot SRTT/2$ data packets, which have high priority, and $S_0 \cdot SRTT/2$ dummy packets out of the $S_0 \cdot SRTT$ dummy packets transmitted during the Detected state. Therefore, the sender must not increase its transmission rate S when it receives the first $S_0 \cdot SRTT/2$ ACKs for dummy packets. In fact, these ACKs cannot be considered as the sign that the loss was due to link errors, i.e., not due to network congestion. Accordingly, $wdsn$ is set to $S_0 \cdot SRTT/2$ in Detected state as shown in Figure 23. As explained in Section 3.3.4, the sender checks the value of $wdsn$ before increasing the data rate upon reception of an ACK for a dummy packet. Hence, this assures TCP-friendliness in case of congestion by preventing the sender to increase its transmission rate when the first $S_0 \cdot SRTT/2$ ACKs for dummy packets are received during the Steady state.

After receiving $S_0 \cdot SRTT/2$ ACKs for dummy segments, the sender increases its transmission rate by $1/SRTT$ each time it receives an ACK for a dummy segment. As a result, if all of the dummy segments transmitted in the Detected state are ACKed to the sender, i.e., the packet loss is due to link error, then the data transmission rate S reaches the value it had before the packet loss was detected, i.e., $S = S_0$. This behavior is further traced in Section 3.4.

In summary, with the help of the ACKs received for the dummy packets transmitted in Detected state, the source can accurately distinguish congestion and link error related packet losses and hence recover from the rate throttle in case of link errors in the Steady state. If the packet loss is due to congestion, then *wdsn* variable assures that RCS maintains TCP-friendliness and does not increase its transmission rate in the Steady state with the reception of ACKs for dummy packets. If the data packet loss is due to link errors, i.e., the network is not congested, then these dummy packets are acknowledged and the data transmission rate, S, is recovered accordingly.

Moreover, it is also likely to experience intermittent link blockages and signal losses in land mobile satellite communication systems due to handoff or signal fading by environmental obstructions such as tunnels, bridges, mountains, and weather patterns such as rainstorms [102]. In such blockage periods, data cannot be successfully transmitted to the receiver and hence performance severely degrades. However, it is essential to make sure that as soon as the blockage period is over the delivery of the multimedia resumes at the highest possible rate, hence minimizing disruption to the on-going connections. In order to achieve this, RCS source incorporates Backoff algorithm whose details are presented in Section 3.3.6.

Therefore, RCS source waits for an ACK during the Detected state to assess the actual reason for the packet loss event. If the packet loss is due to the random link error, then RCS source receives an ACK during the $SRTT$ period of Detected state. If the packet loss is due to congestion, then all TCP friendly protocol sources along the bottleneck would perform rate throttle for congestion resolution in at most one $SRTT$ [48]. Hence, if no ACK is received until the end of one $SRTT$, it can be inferred as an indication of temporary signal

loss instead of the congestion without losing accuracy. In this case, turning back to the Steady state at the end of Detected state is not a good idea. If signal loss is not resolved yet by the end of Detected phase and the source goes into Steady state, it detects another packet loss and turns back to Detected state halving its transmission rate S once again.

This consecutive rate decrease problem can severely degrade the throughput efficiency. In order to avoid this problem, RCS source waits for an ACK during the Detected phase. Therefore, the `ACK_ARRIVAL` event is focused in the `Detected()` algorithm during the Detected phase as shown in Figure 23. The next state is decided according to `ACK_ARRIVAL` event status during this state. If any ACK is received, this means that signal loss is resolved and next state is set to Steady state, i.e., *state*=Steady. If no ACK is received during the Detected phase, then the RCS source does not go back to Steady state, instead it goes to Backoff state, i.e., *state*=Backoff as shown in Figure 20.

3.3.6 Backoff State Operation

In order to prevent unnecessary rate decrease until temporary signal loss is resolved, RCS source backoffs at the end of the Detected phase, i.e., holds its current transmission rate S, if no ACK is received in that phase. Hence, RCS source enters *Backoff* state and calls `Backoff()` algorithm as shown in Figure 24. In this phase, RCS source sends both dummy and data packets with the same transmission rate S. RCS source remains in the Backoff state until it receives any ACK for dummy or data packets. At the end of the Backoff phase, RCS source goes to the Steady state as shown in Figure 20.

In the Backoff state, RCS source executes the `Backoff()` algorithm shown in Figure 24. The Backoff phase starts at t_0 and lasts until any ACK is received. During the algorithm execution,

- The transmission rate, S, and IPG are kept constant.
- The variable $wdsn$ is set to zero $wdsn = 0$.
- The time, t_{next_data}, of the next data packet transmission is set.
- The Boolean variable $ack\_received$ is initially set to false $ack\_received = false$. If

```
Backoff()
    t_0 = t;
    IPG = 1/S;
    wdsn=0;
    ack_received = false;
    t_next_data = t_0;
    while (ack_received == false)
       if (t ≥ t_next_data)
          send(DATA_PACKET);
       if (t ≥ t_next_data + IPG/2)
          send(DUMMY_PACKET);
       t_next_data = t_next_data + IPG;
       if (ACK_ARRIVAL)
          ack_received=true;
    end;
    state=Steady;
end.
```

Figure 24: `Backoff()` Algorithm.

any ACK is received during the Backoff state (`ACK_ARRIVAL`), this variable is set to true, i.e., $ack\_received = true$.

During Backoff phase, RCS source stops halving its transmission rate S and waits for an ACK (`ACK_ARRIVAL`) informing that signal is back. At the same time, RCS source sends data and dummy packets with equal transmission rate S. Once RCS source receives any ACK for either data or dummy packet, it goes into Steady state as in Figure 20. If no ACK is received for a certain timeout period, T_B, the source timeouts and moves back to Initial State, as shown in Figure 20.

The reason for continuing transmission of data packets in Backoff state is two-fold. Since the application is real-time multimedia, which is loss-tolerant to a certain extent and has strict time-deadlines, it is better to keep transmission going on instead of buffering the data until the signal is back. Secondly, while the blockage situation is inferred in the Detected state, it is also possible that the signal may be back during the Backoff state. Hence, instead of waiting until a reception of an ACK ending Backoff state, transmitting loss-tolerant time-sensitive multimedia packets can increase the received service quality.

Furthermore, RCS source sends one dummy packet for each data packet transmitted in Backoff state. The reason for transmitting dummy packets in Backoff state is as follows. Since $wdsn$ is set to 0 as shown in Figure 24, in the Steady state the source increases

its transmission rate S by $1/SRTT$ each time it receives an ACK for the dummy packets transmitted in this phase. Hence, in the following Steady state, the RCS source can increase its transmission rate S immediately with the help of the dummy packets sent in the last $SRTT$ part of the Backoff phase.

3.3.7 Protocol Architecture for Adaptive Real-Time Applications

RCS evaluates the source transmission rate which must not be exceeded to be TCP-friendly. To achieve this goal, RCS must be inserted into an appropriate protocol stack, which is the focus of this section.

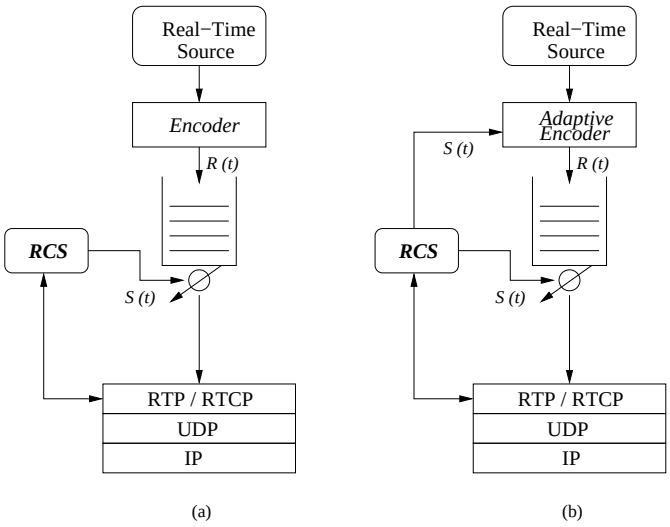

Figure 25: TCP-Friendly Architectures without Adaptive Encoding (a) and with Adaptive Encoding (b).

Consider the system in Figure 25(a). RCS is responsible of calculating at any time, t, the transmission rate, $S(t)$, that the traffic source should respect in order to be TCP-friendly. Since the output rate from the encoder, $R(t)$, is generally different from $S(t)$, i.e., $R(t) \neq S(t)$, a buffer with controlled output rate is inserted between the encoder and the RTP/RTCP layer. Accordingly, the transmission rate is equal to $S(t)$ and the source is TCP-Friendly. However, the above buffer introduces further policing delay, d_π, such that:

- The delay d_π is low when the network is not congested and $S(t)$ is high.
- The delay d_π is high when the network is congested and $S(t)$ is low. Moreover, some

information may get lost due to buffer overflows.

As a consequence, buffering results in increased delay and delay jitter.

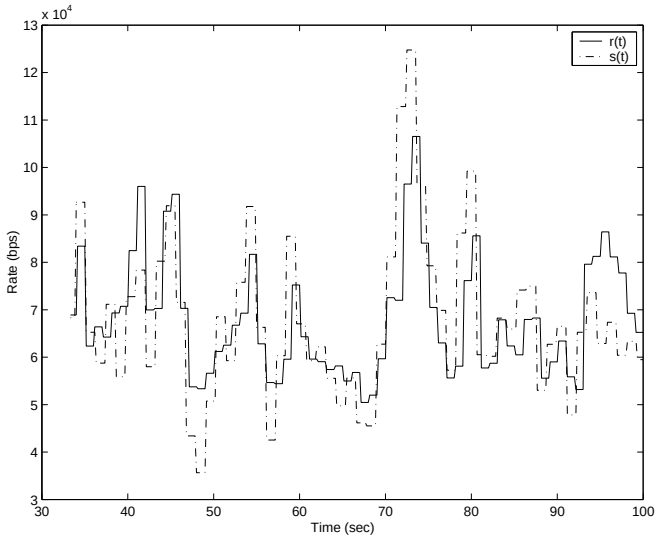

Figure 26: Behavior of $R(t)$ and $S(t)$ with Adaptive Encoding.

Real-time applications are very sensitive to delay and delay jitter thus, when $S(t)$ is low, it is better to decrease the quality of encoding rather than incur in further delays. As a consequence, the encoders with feedback are used as shown in Figure 25(b): RCS provides the encoder with $S(t)$ as input and the encoder adapts its parameters in such a way that $R(t)$ is as close to $S(t)$ as possible. Many adaptive encoding schemes have been proposed in the literature, e.g., [124], [107]. In the ideal case, at any time t, $R(t) = S(t)$ and as a consequence the buffer size, $k(t)$, should be equal to zero as well as the delay $d(t)$, i.e., $k(t) = 0$ and $d(t) = 0$. However, real adaptive encoders are not able to comply exactly with $S(t)$, i.e., in general $S(t)$ and $R(t)$ are different. As an example, in Figure 26 $R(t)$ and $S(t)$ obtained using adaptive video encoding are shown.

The differences between $R(t)$ and $S(t)$ as shown in Figure 26 result in a buffer whose size is given in the upper plot of Figure 27 versus time. In the bottom plot of Figure 27 the delay, $d(t)$, is shown versus time. The values of $S(t)$ have been obtained by simulating RCS in the environment which will be described in Section 3.5.1, whereas the $R(t)$ is the experimental output rate of a MSSG MPEG-2 video encoder [107] when the target source rate is $S(t)$. Both the plots in Figure 27 were obtained considering that the maximum size

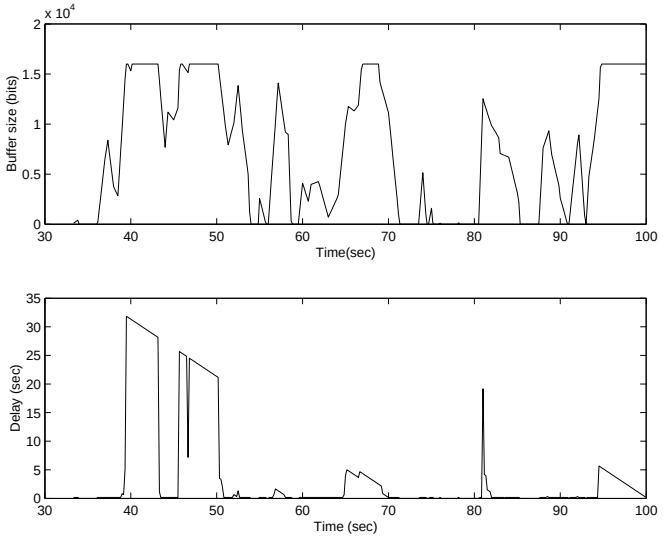

Figure 27: Buffer Size $k(t)$ and Delay $d(t)$ with Adaptive Encoding.

of the buffer is 16kb. Note that the delay can be higher than 30 sec.

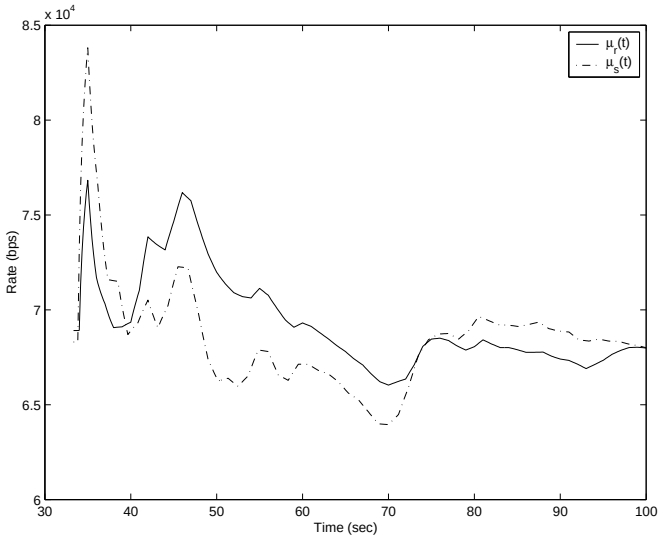

Figure 28: Average rates $\mu_R(t)$ and $\mu_S(t)$ with Adaptive Encoding.

Let us define $\mu_R(t)$ and $\mu_S(t)$ as follows:

$$\mu_R(t) = \frac{1}{t} \cdot \int_0^t R(\tau) d\tau \tag{30}$$

$$\mu_S(t) = \frac{1}{t} \cdot \int_0^t S(\tau) d\tau \tag{31}$$

Note that $\mu_R(t)$ and $\mu_S(t)$ are the average values of $R(t)$ and $S(t)$ in the time interval $[0, t]$. In Figure 28 $\mu_R(t)$ and $\mu_S(t)$ achieved in the case given in Figure 26 are shown. Observe

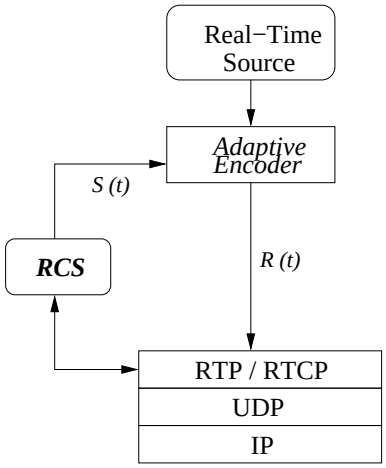

Figure 29: Proposed TCP-Friendly Architecture.

that when t is high, $\mu_R(t) \approx \mu_S(t)$. As a result, if a source generating traffic with rate $S(t)$ is TCP-friendly, a source generating traffic with rate $R(t)$ is TCP-friendly as well. From the previous discussion, the buffer with controlled rate is removed as shown in Figure 29. Using such an architecture, TCP-friendliness can be guaranteed without introducing further delays.

3.4 RCS Behavior

In this section, how the `Detected()`, `Steady()` and `Backoff()` algorithms cooperate when a data packet loss is detected is explored. More in detail, in Section 3.4.1, Section 3.4.2, and Section 3.4.3, RCS behavior is explored when a data packet loss occurs due to link errors, network congestion, and temporal signal loss, respectively.

3.4.1 Packet Loss Due to Link Errors

In this subsection, the behavior of RCS is described on the consequence of a packet loss due to a link error. The RCS source is initially assumed to be at the Steady State at $t = t_0$ as shown in Figure 30. At $t = t_0$, a packet loss is detected and the source moves to Detected state as explained in Section 3.3.5. The detailed trace of RCS protocol operation in case of a packet loss due to link errors is presented as follows.

Let t be time instance in Figure 30.

- $t = t_0$

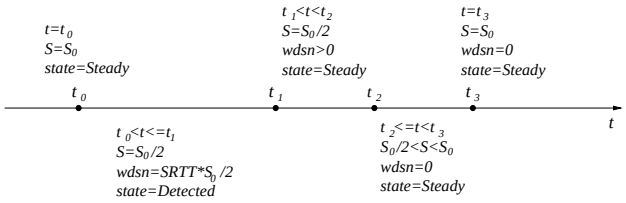

Figure 30: RCS Behavior when a Packet Loss occurs due to Link Errors.

($S = S_0$, state=Steady).

Suppose that the data transmission rate is S_0 at time t_0.

- $t_0 < t \leq t_1$ (where $t_1 = t_0 + SRTT$)

  ($S = S_0/2$, $wdsn = SRTT \cdot S_0/2$, state=Detected).

  Suppose at time t_0 the source detects a packet loss. The source enters the Detected state, halves its transmission rate, i.e., $S = S_0/2$, and sets $wdsn$ to ($wdsn_0 = SRTT \cdot S_0/2$). Moreover, it transmits ($SRTT \cdot S_0$) dummy packets with rate equal to S_0 as explained in Section 3.3.5.

- $t_1 < t < t_2$ (where $t_2 = t_1 + 0.5 \cdot SRTT$)

  ($S = S_0/2$, $wdsn > 0$, state=Steady).

  At time $t = t_1$, the RCS source returns to the Steady state and starts to receive the ACKs for the dummy packets transmitted in the time interval $[t_0, t_1]$. If the network is not congested and thus, dummy packets are not lost, then the number of ACKs for dummy packets received in time interval $t_1 < t < t_2$ is ($SRTT \cdot S_0/2$). Consequently, $wdsn > 0$ and the transmission rate is not increased.

- $t_2 \leq t < t_3$ (where $t_3 = t_1 + SRTT$)

  ($S_0/2 \leq S \leq S_0$, $wdsn = 0$, state=Steady).

  In this time interval, the source receives the ACKs for the other ($SRTT \cdot S_0/2$) dummy packets transmitted during the Detected phase. Since $wdsn$ value is 0, the source can increase its transmission rate by $1/SRTT$ each time it receives an ACK for a dummy packet.

- $t = t_3$

  ($S = S_0$, $wdsn = 0$, state=Steady).

  At time t_3, the source has received the ACKs for all the ($n_{Dummy} = SRTT \cdot S_0$) dummy packets transmitted in the time interval $[t_0, t_1]$. Accordingly, the transmission rate has been increased by ($\Delta S = S_0/2$); in fact

$$\begin{aligned} \Delta S &= (n_{Dummy} - wdsn_0) \cdot \frac{1}{SRTT} \\ &= (SRTT \cdot S_0 - SRTT \cdot S_0/2) \cdot \frac{1}{SRTT} \\ &= \frac{S_0}{2}. \end{aligned} \tag{32}$$

  Consequently, with the help of dummy packet transmission during the Detected state, the RCS source resumes its data transmission rate value it had before the data packet loss was detected.

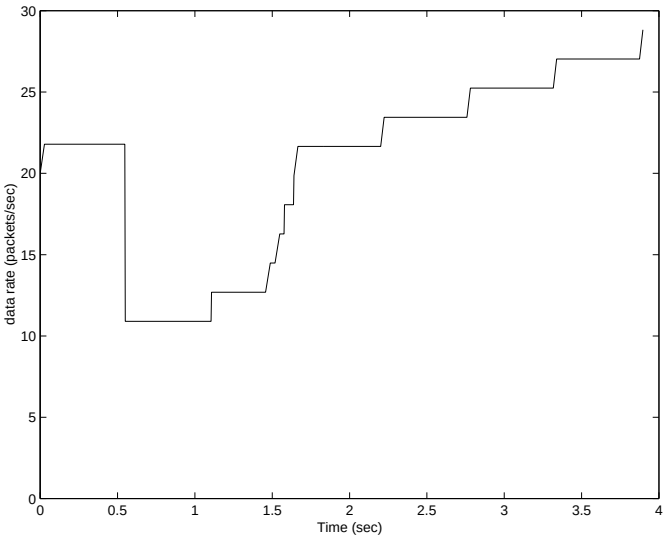

Figure 31: Transmission Rate when a Packet Loss is due to Link Errors.

To observe this behavior, simulation experiments are performed on the network architecture whose details and parameters are explained in detail in Section 3.5. The experiments are performed with assuming $S_0 = 22$ packets/sec and $RTT = 550$ msec. In Figure 31, the transmission rate, S, is shown dependent on time when a packet loss is due to link errors.

As explained in Sections 3.3.4 and 3.3.5, the data rate halved with a detection of a packet loss is recovered by the help of dummy packets transmitted in the Detected state. Hence, this behavior is observed in Figure 31 that the RCS source returns to its previous rate within approximately two round trip times when a packet loss occurs due to link errors.

3.4.2 Packet Loss Due to Network Congestion

Here, the behavior of the RCS algorithms is described when a packet loss occurs due to network congestion. The RCS source is initially assumed to be at the Steady State at $t = t_0$ as shown in Figure 32. At $t = t_0$ a packet loss is detected and the source moves to Detected state as explained in Section 3.3.5. It is shown that RCS source, indeed, halves its transmission rate and hence maintains the TCP-friendly behavior when the network is congested. The detailed trace of RCS protocol operation in this scenario is presented as follows.

Consider a single connection and let t be time instance given in Figure 32.

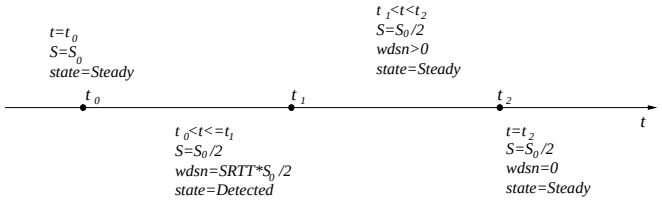

Figure 32: RCS Behavior when a Packet Loss occurs due to Network Congestion.

- $t = t_0$

  ($S = S_0$, state=Steady).

  Let S_0 denote the data transmission rate at time t_0.

- $t_0 < t \leq t_1$ (where $t_1 = t_0 + SRTT$)

  ($S = S_0/2$, $wdsn = SRTT \cdot S_0/2$, state=Detected).

  Suppose that at time t_0 the source detects a packet loss. Also suppose that the above loss is due to network congestion, i.e., the connection path can accommodate at most a transmission rate given by S_0. The source enters the Detected state, halves its

transmission rate, i.e., $S = S_0/2$, and sets $wdsn$ to $wdsn_0 = SRTT \cdot S_0/2$. Moreover, it transmits $(SRTT \cdot S_0)$ dummy packets at a rate equal to S_0. Consequently, the overall transmission rate is $(3 \cdot S_0/2)$. However, the connection path can accommodate at most a rate given by S_0. Since data packets have high priority they are not discarded, whereas the half of the dummy packets (because they have low priority) will be discarded.

- $t_1 < t < t_2$ (where $t_2 = t_1 + SRTT$)

  ($S = S_0/2$, $wdsn > 0$, state=Steady).

  At time $t = t_1$, the RCS source returns to the Steady state and starts to receive the ACKs for the $(SRTT \cdot S_0/2)$ dummy packets transmitted in the time interval $[t_0, t_1]$ which are not discarded by the congested router. Since $wdsn > 0$ in the time interval $[t_1,\ t_2]$, then the transmission rate will not be increased.

- $t \geq t_2$

  ($S = S_0/2$, $wdsn = 0$, state=Steady).

  By the time t_2, all the ACKs for dummy packets which are not dropped by the network reach the source. The value of $wdsn$ has always been higher than zero when the ACKs for dummy packets were received. Consequently, the transmission of the dummy packets during the detected state does not cause any increase in the transmission rate hence RCS preserves TCP-friendliness in case of network congestion.

In Figure 33, the data transmission rate, S, is shown dependent on time when a data packet loss occurs due to network congestion. Figure 33 is obtained through simulation assuming $S_0 = 22$ packets/sec and $RTT = 550$ msec. At time $t_0 = 0.5$ sec, a packet loss is detected, accordingly, the transmission rate, S, is set to $S = 11$ packets/sec. As explained in Section 3.3.5, by the help of protocol variable $wdsn$, the RCS source does not increase the rate for the first half of the dummy ACKs it receives during the Steady state. Since the congestion is experienced, the second half of the dummy packets transmitted are discarded and hence no rate increase is performed with the reception of dummy ACKs during the

Steady state. Therefore, for $t > t_0$, the transmission rate increases by {one packet}/RTT each round trip time as in the TCP [64] case, i.e., RCS behavior is TCP-friendly.

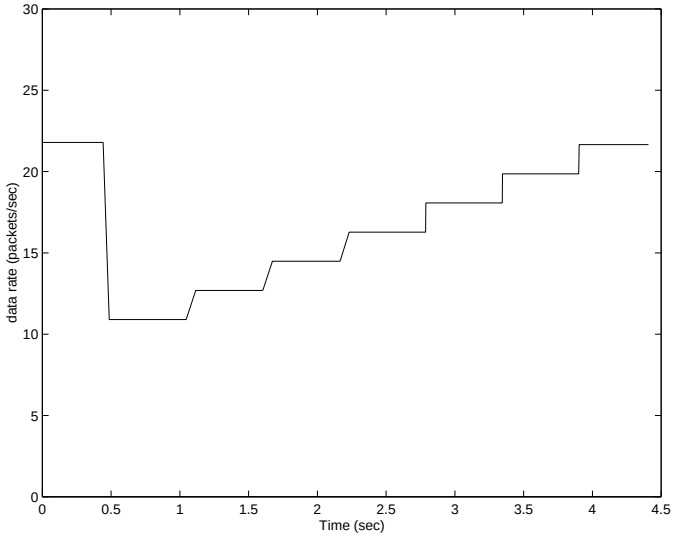

Figure 33: Transmission Rate when a Packet Loss is due to Network Congestion.

3.4.3 Packet Loss Due to Temporary Signal Loss

Here, it is assumed that the packet loss occurs due to temporary signal loss and describe the resulting behavior of the proposed protocol. The RCS source is initially assumed to be at the Steady State at $t = t_0$ as shown in Figure 34. At $t = t_0$ a packet loss is detected and the source moves to Detected state as explained in Section 3.3.5.

Let t be time instance in Figure 34.

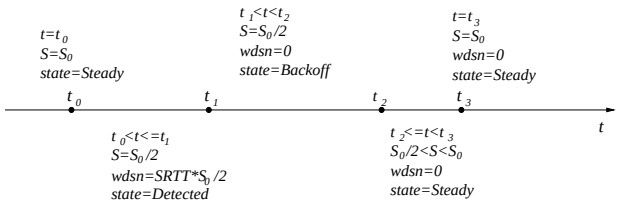

Figure 34: RCS Behavior when a Packet Loss occurs due to Temporary Signal Loss.

- $t = t_0$

  ($S = S_0$, state=Steady).

  Suppose that the data transmission rate is S_0 at time t_0.

- $t_0 < t \leq t_1$ (where $t_1 = t_0 + SRTT$)

  ($S = S_0/2$, $wdsn = SRTT \cdot S_0/2$, state=Detected).

  Suppose at time t_0 the source detects a packet loss due to temporary signal loss. The source enters the Detected state, halves its transmission rate, i.e., $S = S_0/2$, and sets $wdsn$ to ($wdsn_0 = SRTT \cdot S_0/2$).

- $t_1 < t < t_2$ (where t_2 is the time when the first ACK is received).

  ($S = S_0/2$, $wdsn = 0$, state=Backoff).

  Due to the signal loss, source does not receive any ACK during this period and goes to Backoff state at $t = t_1$, i.e., *state*=Backoff. Source transmits data and dummy packets with rate $S = S_0/2$. Once an ACK is received, source terminates Backoff phase. Since the resolution of signal loss is detected with ACK reception, $SRTT \cdot S_0/2$ dummy packets are transmitted before Backoff phase is terminated.

- $t_2 \leq t < t_3$ (where $t_3 = t_2 + SRTT$)

  ($S_0/2 \leq S \leq S_0$, $wdsn = 0$, state=Steady).

  During this interval, the source receives the ACKs for the dummy packets transmitted during the Backoff phase after the temporal signal loss is resolved. Since $wdsn$ value is 0, the source can increase its transmission rate by $1/SRTT$ each time it receives an ACK for a dummy packet.

- $t = t_3$

  ($S = S_0$, $wdsn = 0$, state=Steady).

  At time t_3, the source has received the ACKs for all the ($SRTT \cdot S_0/2$) dummy packets transmitted in the time interval $[t_1, t_2]$. Hence, the original transmission rate S_0 is resumed in approximately $SRTT$ period after the temporal signal loss is resolved.

To illustrate the enhanced behavior of RCS in temporal signal loss conditions, the transmission rate change, S, is shown with time in Figure 35. Figure 35 is obtained by simulation assuming $S_0 = 22$ packets/sec and $RTT = 550$ msec. The duration of signal loss $D_{SignalLoss}$

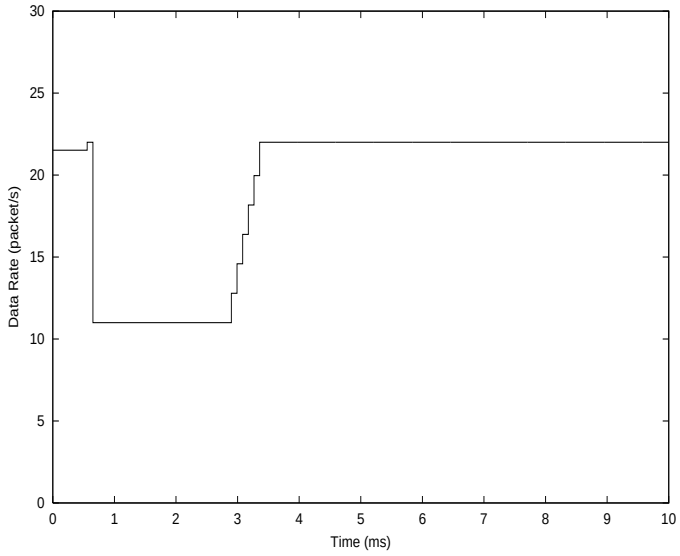

Figure 35: Transmission Rate when a Packet Loss is due to Temporal Signal Loss.

is assumed to be $2 \cdot SRTT$. In this case, RCS source transmits dummy and data packets in the Backoff state with the same rate for the reasons explained in Section 3.3.6. Hence, RCS source avoids unnecessary consecutive rate decrease in case of signal loss and resumes its original transmission rate S_0 within approximately one $SRTT$ after the signal loss is resolved.

3.5 Performance Evaluation

In order to investigate the performance of the RCS, extensive simulation experiments are conducted using ns-2 [111] [1]. The details of the simulation environment are presented in Section 3.5.1. Then, RCS throughput, dummy traffic overhead, fairness, the performance achieved in case of temporal link loss conditions, and the jitter performance are evaluated in Sections 3.5.2, 3.5.3, 3.5.4, 3.5.5, and 3.5.6 respectively.

3.5.1 Simulation Environment

RCS is intended to achieve high throughput and TCP-friendly rate control for real-time traffic in networks with long propagation delays and lossy links. Satellite networks are

[1]The RCS ns-2 agent can be downloaded from the following URL `http://www.ece.gatech.edu/research/labs/bwn`

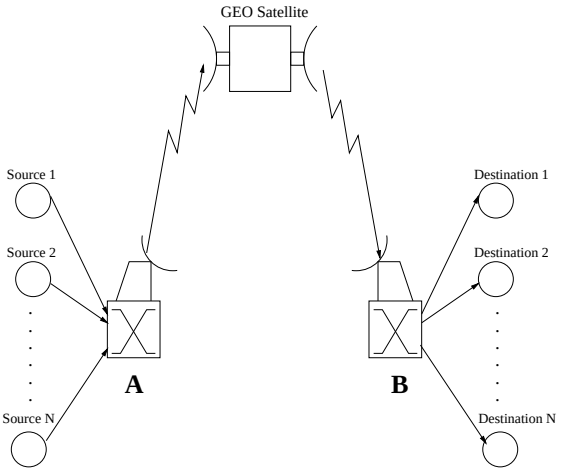

Figure 36: Simulation Scenario.

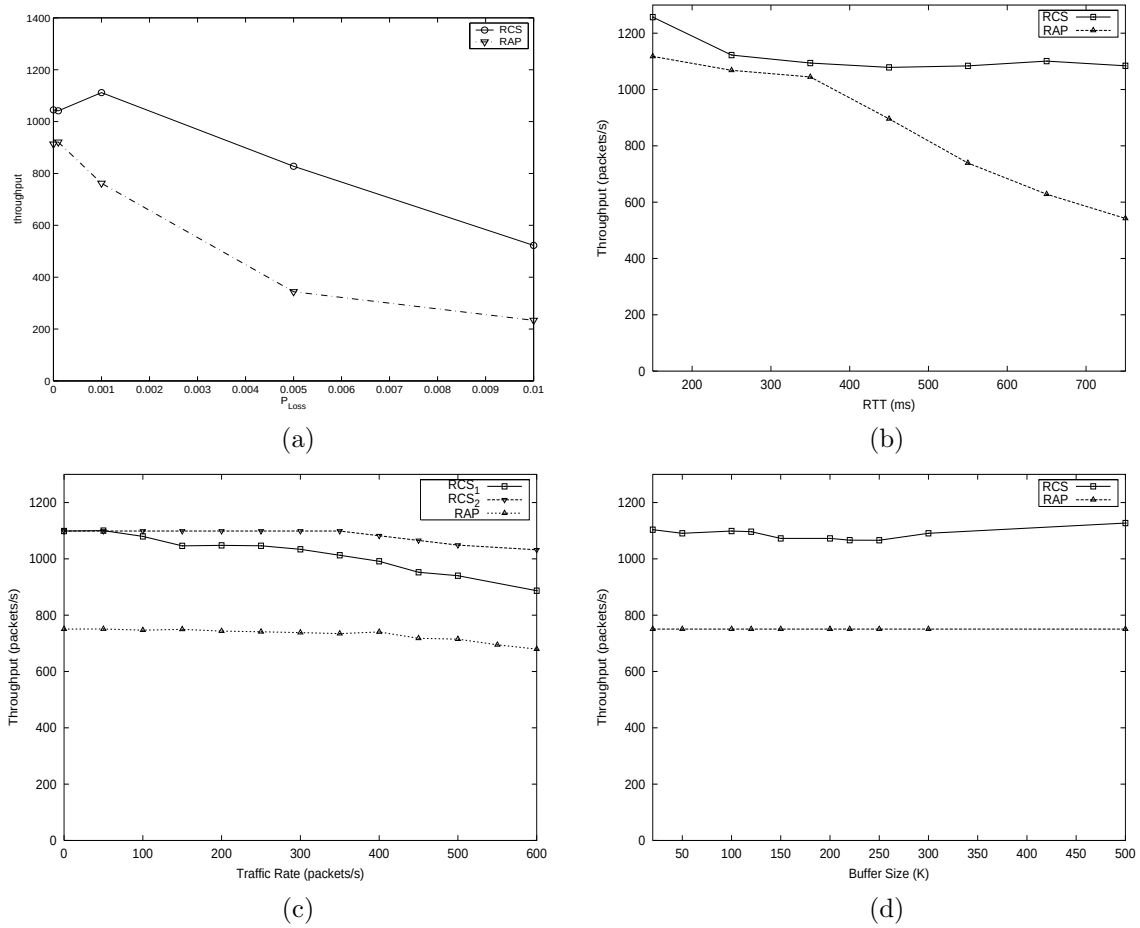

Figure 37: Throughput Performance of RCS (solid lines) and RAP (dashed lines) for (a) varying loss probability P_{Loss} (b) varying round-trip time RTT (c) varying background low-priority traffic rate S_L (d) varying buffer size K.

typical examples of networks with high bandwidth-delay products and high bit error rates. Therefore, the network topology shown in Figure 36 is simulated where N sources transmit data to N destinations over lossy and high propagation delay Geostationary (GEO) satellite links. The N streams are multiplexed in the router A, whose buffer accommodates K packets. The GEO satellite is assumed to be a *bent-pipe satellite*, i.e., it directly relays the packets without any buffering and processing. The multiplexed streams are de-multiplexed in the router B and then routed to corresponding N destinations. Both data and dummy packets may get lost due to link errors with probability P_{Loss} in the lossy link. It is assumed that $N = 10, K = 50$ packets and the link capacity is $c = 1300$ packets/sec, which is approximately equal to 10 Mb/sec for packets of length 1000 bytes. As an example of a long delay link, unless otherwise stated, it is assumed that the RTT value is $RTT = 550$ msec, which is a typical RTT value in the GEO satellite links [10]. Further details of the simulation environment corresponding to specific experiments are given in each of the following performance evaluation subsections.

3.5.2 Throughput Performance

In order to assess the throughput performance of the RCS scheme, extensive simulation experiments are performed using the simulation scenario described in Section 3.5.1. Throughput performance is evaluated for different values of the packet loss probability P_{Loss}, the round trip time, RTT, the rate of the background low-priority traffic S_L, and the buffer size K.

The results obtained are shown in Figure 37. For the sake of comparison, the performance achieved by the *rate adaptation protocol* (RAP) [88] is also shown.

In Figure 37(a), RCS throughput is shown for different values of loss probabilities, ${P_{Loss}}^2$. Here, it is assumed $RTT = 550$ ms. As shown in Figure 37(a), RCS significantly improves the throughput performance over RAP for all values of P_{Loss}. RCS throughput is maximum when $P_{Loss} = 10^{-3}$. With increasing P_{Loss}, the throughput of both RCS and RAP

[2]The bit error rate (BER) in satellite networks can be as high as 10^{-4}, i.e., one bad bit out of 10000 bits. For packets of 1000 bytes, the BER 10^{-4} gives a packet loss probability, P_{Loss} higher than 10^{-2} even if powerful FEC algorithm is applied.

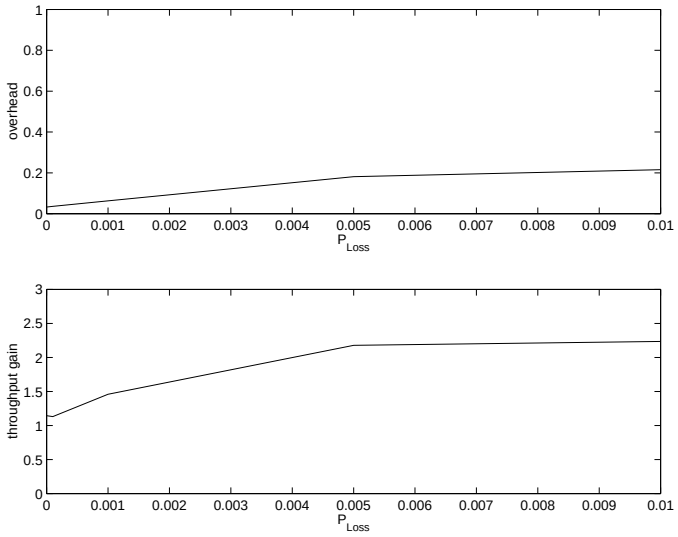

Figure 38: Comparison of Bandwidth Overhead and Throughput Gain between RCS and RAP.

decreases since they both follow the conservative rate decrease behavior to preserve TCP-friendliness. However, the throughput degradation RAP experiences with increasing P_{Loss} is much more serious than RCS and the throughput obtained by using RCS is always higher than the throughput obtained by using RAP [88]. For $P_{Loss} = 5 \cdot 10^{-3}$, RCS achieves more than 150 % throughput improvement over RCS. This is mainly because the RCS algorithms efficiently distinguishes the cause of the packet loss and recovers from the unnecessary rate throttles by the help of dummy packets.

In Figure 37(b), RCS throughput is shown for varying round-trip time values, i.e., 100 ms $\leq RTT \leq$ 750 ms. Here, it is assumed $P_{Loss} = 10^{-3}$. RCS throughput is not significantly affected by increasing RTT while RAP throughput severely degrades for high RTT values, i.e., for $RTT \geq 250$ ms. Such serious performance loss experienced by RAP is due to the amplifying effects of the high propagation delay on the performance degradation due to link errors. On the contrary, as explained in Section 3.4.1, RCS can quickly recover from the rate halving it performs in case of packet loss due to link errors by the help of dummy packets. Therefore, RCS achieves high throughput performance in links with high bit error rates and high propagation delays. Furthermore, note that RCS also outperforms RAP for moderate RTT values such as $RTT = 150$ ms as shown in Figure 37(b). Therefore, while

the algorithms of RCS are specifically tailored to address the challenges in networks with high bit error rates and high bandwidth-delay products, high throughput performance can still be achieved in case of moderate RTT values.

In order to investigate the effects of the low-priority background traffic on the RCS performance, it is now assumed that low-priority background traffic, which is generated by an application at the data rate of S_L packets/s, flows through the satellite channel given shown in Figure 36. Two scenarios are considered. In the first one, the low-priority background traffic sources are generating UDP traffic at a fixed data rate S_L without performing any congestion control. In the second scenario, it is assumed that the background low-priority traffic sources are TCP-friendly and perform congestion control. Here, it is assumed that $P_{Loss} = 10^{-3}$ and $RTT = 550$ ms and the same simulation environment described in Section 3.5.1 is used.

For the first scenario, a wide range of S_L values, i.e, $0 \leq S_L \leq 600$ packets/s, is considered. As shown in Figure 37(c), the throughput performance of RCS (RCS_1 curve) is not significantly affected by the low-priority traffic for moderate background traffic rates, i.e., $S_L = 400$ packets/s. For $S_L > 400$ packets/s, RCS throughput degrades slightly. This is mainly because the low-priority background traffic in this scenario does not perform congestion control and hence creates congestion in the network. Because of very high low-priority traffic rate, higher number of dummy packets, which are transmitted by the RCS for link probing, are also dropped at the routers. However, as it is observed in Figure 37(c), this degradation is very slight even for $S_L = 600$ packets/s and RCS always outperforms RAP for all values of S_L.

In the second scenario, simulation experiments are performed using the TCP-friendly sources generating low-priority background traffic in the scenario given in Section 3.5.1. RAP is used as a rate control protocol at the background traffic sources to generate responsive low-priority traffic. As shown in Figure 37(c), the throughput of RCS (RCS_2 curve) is not affected by the background traffic and remains almost constant. This is because the low-priority traffic also performs rate control in case of congestion and hence it does not significantly affect the dummy traffic. As a result, low priority background traffic

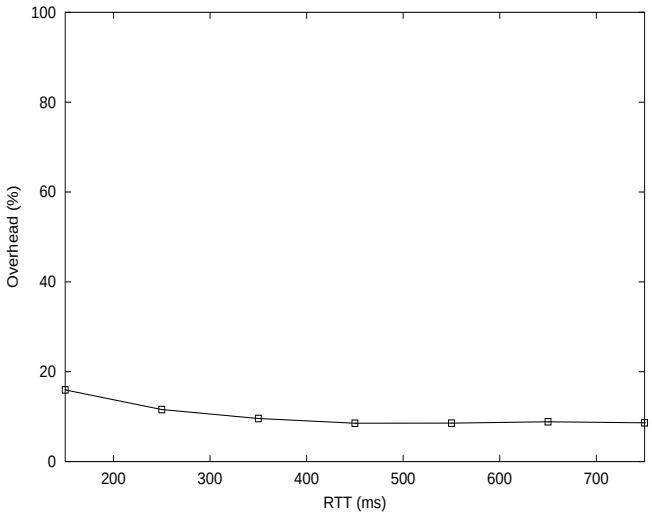

Figure 39: The overhead due to dummy packet traffic for varying round-trip time RTT.

does not cause significant degradation in the accuracy of the RCS algorithms and hence RCS maintains its high throughput performance for various low priority background traffic scenarios.

Furthermore, the throughput performance is investigated for varying buffer size K. It is assumed that $P_{Loss} = 10^{-3}$, $RTT = 550$ ms, $20 \leq K \leq 500$ and the same simulation environment described in Section 3.5.1 is used. As shown in Figure 37(d), RCS throughput does not change significantly for $N \leq 300$. However, for higher buffer sizes, RCS throughput slightly increases. This is mainly because of the rate-based nature of the RCS. Since RCS can recover from the rate throttle due to link errors, it can more efficiently utilize link resources in the links with high bit error rates and high propagation delays. Thus, an increase in the buffer size can further contribute to this link utilization efficiency. Note also that RCS outperforms RAP for all values of buffer sizes.

3.5.3 Dummy Packet Traffic

In this section, the amount of traffic generated by the transmission of dummy packets is analyzed. Simulation experiments are performed with the scenario described in Section 3.5.1.

In Figure 38, the overhead due to the transmission of dummy packets is shown. Let

- N_{Dummy} be the total number of transmitted dummy packets.
- N_{Data} be the total number of transmitted data packets.

The overhead is defined as:

$$overhead = \frac{N_{Dummy}}{N_{Data} + N_{Dummy}}. \tag{33}$$

In the upper plot of Figure 38, the overhead is shown to be dependent on the loss probability, P_{Loss}. Obviously, the overhead increases when P_{Loss} increases. This is mainly because as P_{Loss} increases higher number of packet losses are detected and hence RCS sources enter Detected state more frequently. This leads to transmission of higher number of low priority dummy packets as explained in Section 3.3.5. Note that the overhead can be as high as 21.5% when $P_{Loss} = 10^{-2}$. However, using dummy packets RCS achieves much higher throughput than other rate control schemes for real-time traffic. For example, in the bottom plot of Figure 38, the throughput gain obtained using RCS is shown. The throughput gain is evaluated as the ratio between the throughput achieved by RCS and the throughput achieved by RAP. Note that when $P_{Loss} = 10^{-2}$, the overhead is 21.5%, but the throughput gain is higher than 200%.

Furthermore, note that low priority dummy packets can carry new data as well. Since real-time multimedia flows are time-sensitive and loss-tolerant to a certain extent, low-priority dummy packets can carry actual data payload. In this case, the bandwidth consumed by the low-priority dummy packets does not stand as overhead, instead, it contributes to the service received at the sink.

Simulation experiments are also performed to investigate the effects of RTT on the amount of created dummy packet traffic. As shown in Figure 39(a), the overhead created by the dummy traffic is higher for low RTT values, i.e., $RTT \approx 100$ms. As RTT increases, the overhead drops to as low as 10 % for $RTT \geq 400$ ms. This is because the frequency of entering Detected state and hence the number of dummy packets transmitted decrease with increasing RTT. This result is also consistent with the fact that RCS is mainly developed for the links with high bit error rates and high propagation delay links.

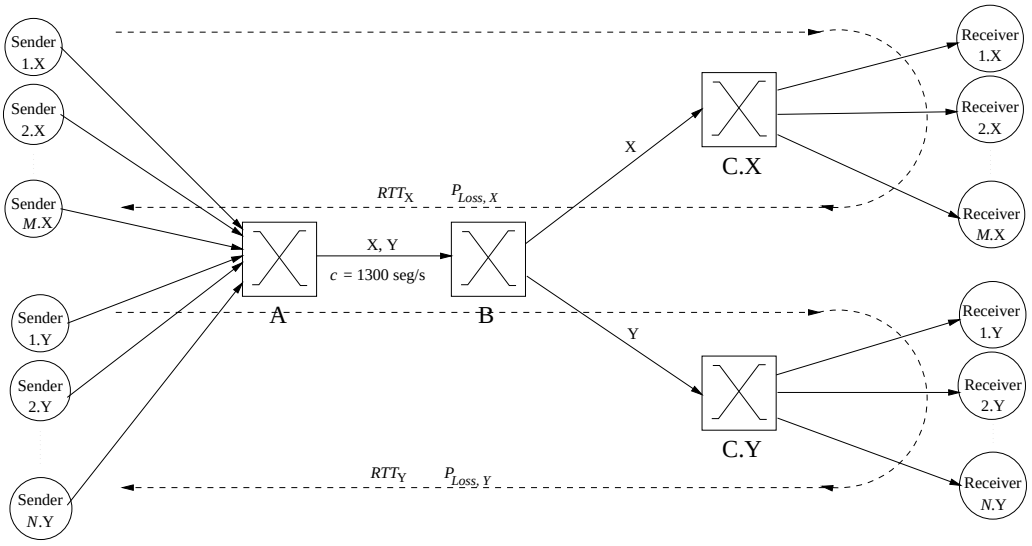

Figure 40: Network Model for the Fairness Evaluation in the Heterogeneous Cases.

3.5.4 Fairness

For the fairness performance of RCS protocol, two different scenarios, i.e., homogeneous and heterogeneous fairness, are considered. For the homogeneous case, the fairness among RCS flows in sharing the same bottleneck is observed. For heterogeneous case, the fairness of RCS protocol is explored in heterogeneous scenarios, i.e., connections can use different protocols or flow through different network paths.

3.5.4.1 Homogeneous Scenario

In this scenario, all connections pass through the same path and run RCS protocol in the simulation environment described in Section 3.5.1. Let $acked_i(t)$ represent the number of data packets acknowledged in the time interval $[0, t]$ for connection i, for $i = 1, 2, ..., N$. In Figure 41, $acked_i(t)$ is shown dependent on time t for $i = 1, 2, ..., N$. It is seen that at any time, t, all connections have been acknowledged the same number of data packets approximately, i.e., $acked_{i'}(t) \approx acked_{i''}(t)$, for any i' and i''. This means that each RCS connection is given a fair share of network resources. Here, it is assumed that $N = 10$, $K = 50$ packets, $c = 1300$ packets/sec, $P_{Loss} = 0$ and $RTT = 550$ msec. Note that similar results are obtained for several other cases.

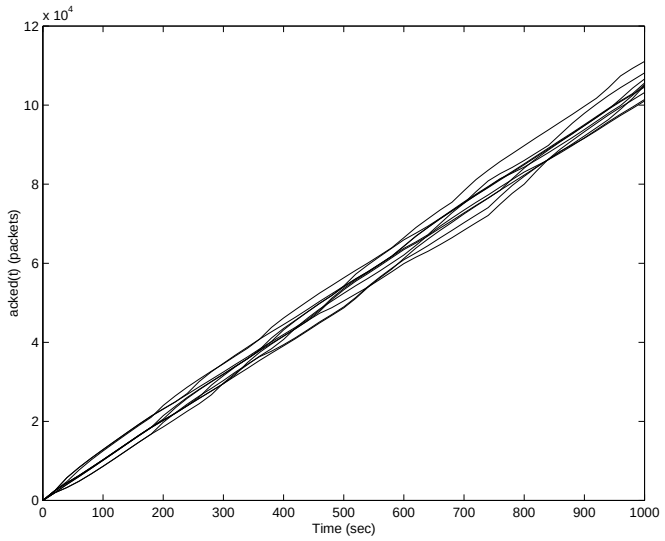

Figure 41: Fairness in the Homogeneous Case.

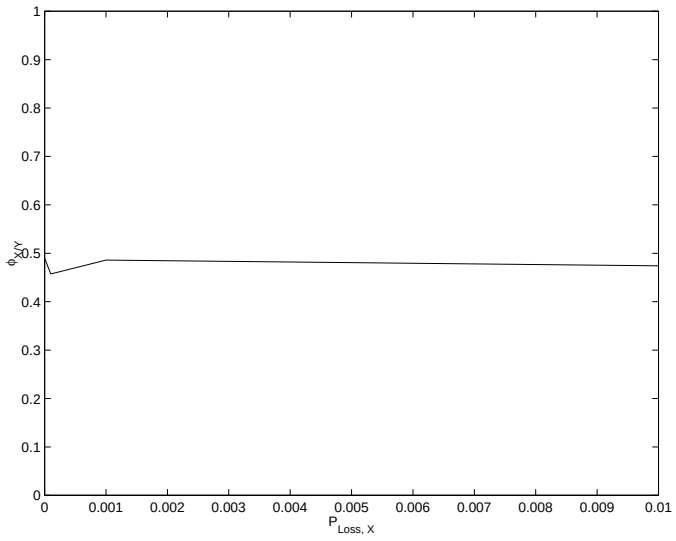

Figure 42: Fairness between RCS Connections with Different Paths.

3.5.4.2 Heterogeneous Scenario

To investigate the fairness of RCS in heterogeneous scenarios, the simulation scenario shown in Figure 40 is considered. In this case, there are M connections of type X and N connections of type Y. Connections of type j, for $j = X, Y$, are characterized by round-trip time equal to RTT_j, and loss probability for link errors equal to $P_{Loss,j}$. All connections pass through the link connecting the routers A and B, which is assumed to be the bottleneck and whose capacity is assumed to be $c = 1300$ packets/sec. The fairness, ϕ, is the ratio between the

average throughput of connections of type X, r_X, and the average throughput of connections of type Y, r_Y, i.e.,

$$\phi = \frac{r_X}{r_Y} \tag{34}$$

It is obvious that the fairness becomes higher as ϕ approaches 1.

In the first experiment scenario, the fairness is investigated when connections X and Y use RCS and pass through lossy links with different propagation delays, i.e., $RTT_X = 550$ msec, $RTT_Y = 250$ msec. Here, it is assumed that $N = M = 5$. As shown in Figure 42, the ratio of the average throughput achieved by the connections X and connections Y is approximately equal to the inverse ratio of their corresponding RTTs, that is,

$$\phi \approx \frac{RTT_Y}{RTT_X} \tag{35}$$

This is because the throughput of the TCP-friendly AIMD schemes are inversely proportional to RTT as it has already been observed and analyzed in the literature [72, 116].

In the second experiment scenario, it is assumed that both connections X and Y pass through the same type of lossy and high propagation delay link, i.e., $RTT = 550$ ms, but connections of type X are TCP connections and connections of type Y are RCS connections. Moreover, it is assumed that $N = M = 5$. In Figure 43, it is seen that r_Y values are higher than r_X thus, resources are not shared equally between TCP and RCS connections. This is mostly due to the problems of TCP in networks with high bandwidth-delay product and high link error rate [85]. In fact, in Figure 43, it is shown that $r_X \approx r_{TCP}$, where r_{TCP} is the average throughput when all connections, i.e., both X and Y connections, use TCP. This means that RCS significantly improves network efficiency without penalizing TCP flows and the degradation of the TCP share is mainly because of its shortcomings when applied to the networks with high bit error rates and high propagation delays as discussed in Section 3.2.

Hence, in order to further investigate the fairness of the RCS to TCP connections, simulation experiments are performed using the simulation environment shown in Figure 40 with different configurations. In this scenario, it is assumed that connections X, i.e., RCS sources, are connected to router A via geostationary satellite links and connections Y,

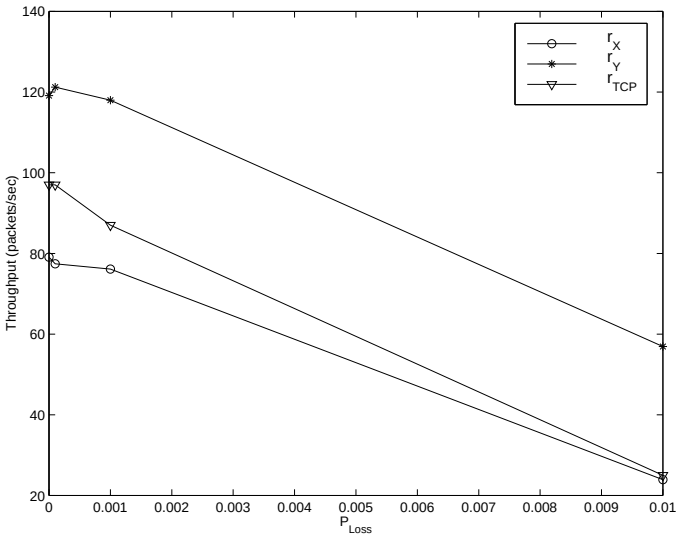

Figure 43: Fairness between RCS and Traditional TCP Connections [14].

i.e., TCP sources, are connected to router A via error-free wired links. Thus, it is assumed that $RTT_X = 550$ ms, $RTT_Y = 110$ ms, $P_{Loss,X} = 10^{-3}$, $P_{Loss,X} = 0$, and $N = M = 5$. The objective of such simulation configuration is to assess the fairness between RCS and TCP sources which are not experiencing link errors or high propagation delays.

To assess the fairness performance of RCS, *Jain's fairness index* [35] is used which quantifies the degree of similarity between the amount of link resources used by all connections. Let x_i be the throughput of the i^{th} connection and let n be the number of connections competing for the same bottleneck resources. Then, the *Jain's Fairness Index*, ψ, can be evaluated as follows:

$$\psi = \frac{(\sum_{i=1}^{n} x_i)^2}{n \cdot \sum_{i=1}^{n} x_i^2} \tag{36}$$

As it can be observed from (36), ψ is always between 0 and 1. When the fairness index is 1, all of the connections consume the same amount of bottleneck resources. The fairness index ψ decreases as the difference between the throughput values achieved by different connections increases.

In the first set of experiments under the simulation scenario described above, the fairness index of RCS vs. P_{Loss_X} is explored. As it is observed in Figure 44(a), RCS fairly shares the link resources with TCP sources for $P_{Loss_X} < 0.001$, i.e., $\psi \approx 1$. For $P_{Loss_X} \geq 0.001$, the fairness index decreases as the throughput performance of the RCS degrades with increasing

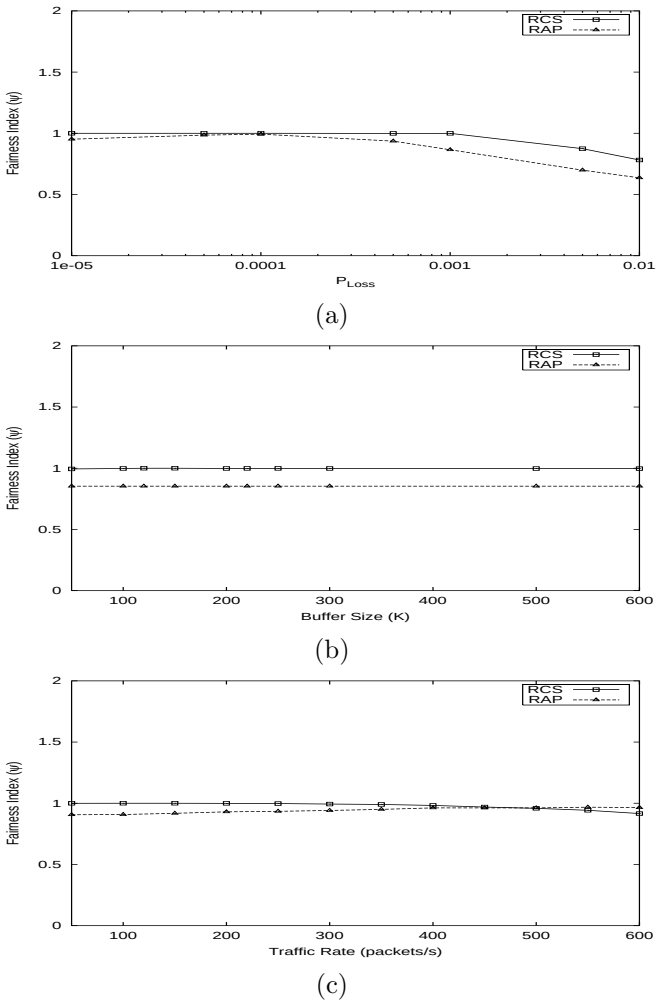

Figure 44: Fairness Index ψ between RCS and TCP connections, where TCP sources do not experience the same link conditions with high bit error rates and high propagation delays, for (a) varying packet loss probability P_{Loss} (b) varying buffer size K (c) varying background low-priority traffic rate S_L.

P_{Loss_X} as also observed in Section 3.5.2. The same experiments are also performed with RAP sources instead of RCS. Note that the fairness index obtained with RAP sources is lower than RCS for $P_{Loss_X} \geq 0.001$ as shown in Figure 44(a). This is again because of the fact that RAP cannot adapt itself to high bit error rate conditions and hence experience throughput degradation as observed in Section 3.5.2.

Under the same simulation scenario, the effects of buffer size K on the fairness of the RCS to the TCP sources is also explored. As shown in Figure 44(b), all of the connections consume almost the same amount of bottleneck resources for all buffer sizes, i.e., $\psi \approx 1$ for

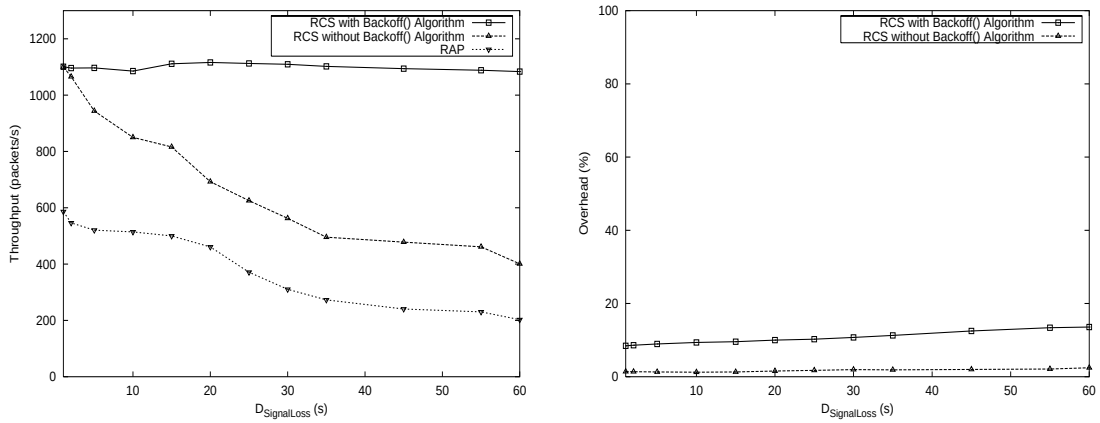

Figure 45: (a) Throughput Performance of RCS with and without `Backoff()` algorithm and RAP for Different Values of Signal Loss Duration, $D_{SignalLoss}$. (b) The amount of dummy packet traffic in temporal link loss conditions for different values of $D_{SignalLoss}$.

$K \leq 600$. This is mainly because the RCS throughput is not significantly affected by the buffer size as also observed in Figure 37(d). Note that the same behavior is also observed for the experiments performed with RAP sources as shown in Figure 44(b). However, note also that the fairness index is lower than RCS case, which is again due to the degraded throughput performance of RAP protocol in the links with high propagation delays and high bit error rates.

Furthermore, the effects of background low-priority traffic on the fairness performance of the RCS is explored. In this scenario, the UDP sources create background low-priority traffic with constant data rate S_L as in Section 3.5.2. The simulation experiments are performed for $0 \leq S_L \leq 600$ packets/s. As shown in Figure 44(c), RCS link share does not exceed that of TCP for $S_L > 0$. Furthermore, RCS link share decreases with increasing S_L and hence the fairness index ψ slightly decreases as shown in Figure 44(c). This is because the RCS throughput slightly decreases with increasing low-priority background traffic rate as it was also observed in Section 3.5.2. However, in this case, the fairness index achieved by RAP sources does not vary with varying background low-priority traffic rate as shown in Figure 44(c). This is because RAP protocol does not utilize link probing technique based on low-priority packet and hence its throughput is not affected by the background low-priority

traffic.

As a result, RCS achieves high throughput performance in the networks with high bit error rates and high bandwidth-delay products while it preserves fairness to the TCP sources sharing the same bottleneck.

3.5.5 Temporal Link Blockage

In order to observe the performance of RCS in case of temporal signal loss conditions, the same network topology given in Section 3.5.1 is simulated with varying signal loss durations $D_{SignalLoss}$. Here, it is assumed that $P_{Loss} = 10^{-3}$ and $RTT = 550$ ms. In Figure 45(a), the throughput of RCS and RAP [88] is shown for signal loss of duration $0 \leq D_{SignalLoss} \leq 60$ seconds. Note that while RAP performance throughput decreases dramatically as the signal loss duration increases, the performance degradation of RCS is negligible.

In order to assess the advantages provided by the Backoff algorithm, in Figure 45(a), the performance of RCS is also shown if the Backoff algorithm is not implemented. The Backoff algorithm gives significant performance increase because it avoids additional transmission rate throttles until an ACK is received indicating that the signal has been recovered. Furthermore, the ACKs received for the dummy packets transmitted during the Backoff state help RCS source to immediately increase its data rate as explained in Section 3.3.6. At $D_{SignalLoss} = 30$ seconds, RCS outperforms RAP by more than 260% throughput improvement.

Since the Backoff algorithm achieves this throughput improvement in case of blackouts by transmitting dummy packets until the link is back, the overhead caused by the dummy packet traffic is evaluted for varying $D_{SignalLoss}$. As shown in Figure 45(b), the overhead due to low-priority dummy packets slightly increases with increasing duration of temporal link loss. This is because RCS stays in the Backoff state and continuously transmits dummy packets until the link is back as explained in Section 3.3.6. However, by means of the additional traffic caused by the dummy packets transmitted during Backoff state, the throughput gain achieved in case of temporal signal loss situations is very significant as shown in Figure 45(a).

3.5.6 Jitter Performance

The jitter experienced by RCS is shown in the upper plot of Figure 46. For the sake of comparison in the bottom plot, the jitter experienced by RAP [88] is also shown. The peaks shown in both plots are due to the rate throttles which are triggered when packet losses are detected.

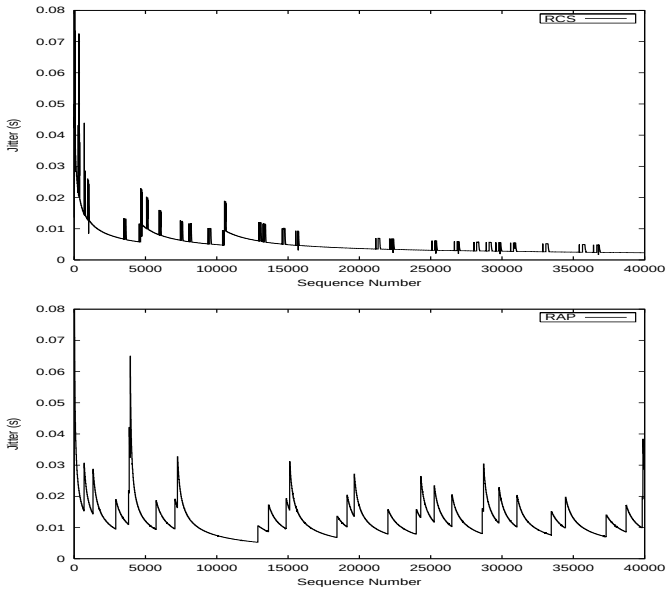

Figure 46: Jitter experienced by each packet sent by RCS and RAP sources.

Although RCS sources experience slightly reduced jitter, note that the delay-jitter suffered by RCS and RAP sources are mostly similar. This is not surprising because the delay-jitter performance mainly depends on the variation of the delay suffered by packets in the network, which cannot be controlled by end-to-end rate control mechanisms such as RCS and RAP.

CHAPTER IV

ADAPTIVE TRANSPORT LAYER FOR NEXT GENERATION WIRELESS INTERNET

In this chapter, a unified adaptive transport layer (ATL) suite is introduced for NGWI. ATL incorporates a new adaptive transport protocol, TCP-ATL, for reliable data transport; and a new adaptive rate control protocol, RCP-ATL, for multimedia delivery in the NGWI. The ATL suite was presented in [7]. A review of related work in transport layer protocols for wireless systems and their inadequacies in addressing the diversity in the NGWI are explored in Section 4.2. The effects of the architectural heterogeneity on the performance are investigated via a case study; and the adaptive congestion control methods for both TCP-ATL and RCP-ATL are then developed in Section 4.3. The overview of ATL protocols along with their detailed operations are explained in Section 4.4. The performance evaluation of TCP-ATL and RCP-ATL and the simulation results are presented in Section 4.5.

4.1 Motivation

The Next Generation Wireless Internet (NGWI) is expected to provide a wide range of services including high speed data and real-time multimedia to mobile users. To realize this expectation, a diverse set of challenges need to be addressed, which are posed by heterogeneous wireless networking environments within NGWI and the according application requirements. These significantly challenging heterogeneities imposed by the NGWI can be outlined as follows:

- **Heterogeneous Wireless Architectures:** The wireless systems that will be part of the NGWI have different characteristics summarized as follows:

- *Access Delay*: While the one-way access delay in the wireless link may not be significant in WLAN, a typical round-trip time (RTT) varies between few hundred milliseconds and one second in 3G links due to the extensive physical layer processing, e.g., forward error correction (FEC), interleaving and transmission delays [60]. The access delay is much higher in satellite links, which have high propagation delay up to 270 ms [10].
- *Link Errors*: The conventional reliable Transmission Control Protocols (TCP) vastly used for reliable data transport presume the existence of an underlying reliable physical link. Thus, they invoke congestion control in case of wireless link related packet losses. This leads to significant performance degradation, whose severity is proportional to the wireless link conditions, i.e., bit error rate and delay. The packet loss rates vary from very low levels in near-wireless environments such as WLAN, 3G pico-cells to higher than 1% in macro-cellular environments and satellite networks [106]. The mobility related packet losses, i.e. blackouts due to handoff or signal loss, amplify the degradation.
- *Mobility Pattern*: The mobility rate may increase the number of blackouts due to handoff and hence decrease the transport efficiency. During a connection period, almost no handoff is experienced in the global coverage, whereas frequent handoffs may take place in pico-cellular environments.

- **Heterogeneous Service Demands:** The services that will be provided by the NGWI vary from high rate reliable data to real-time multimedia such as live video streaming. While the former requires 100% reliable transport, the latter, instead, needs timely delivery and smooth rate variation. For multimedia traffic delivery, UDP-based streaming without any transmission rate control may also lead to unfairness to the TCP sources and further result in a congestion collapse [52].

The architectural heterogeneities must be captured dynamically while mobile users may roam during their connection duration. Current existing transport layer protocols have been developed for a specific network paradigm in mind, e.g., for Wireless LANs (WLAN),

micro/macro wireless systems or for satellite systems. Using these existing different transport layer protocols for NGWI to support global roaming of mobile users is not a practical solution due to processing and memory constraints of wireless terminals. Thus, there is a need for a unified adaptive transport layer protocol suite which can address the architectural heterogeneities for roaming mobile users and achieve the best performance for NGWI.

All these heterogeneities coupled with the processing and power limitations of wireless terminals call for a unified, efficient, and seamless *adaptive transport layer*. In order to address this need, a new adaptive transport layer (ATL) for the Next Generation Wireless Internet is presented in this chapter.

ATL is a unified adaptive transport layer that *dynamically adjusts its protocol configurations to adapt to heterogeneous wireless environments and supports reliable data and multimedia delivery*. Some of its salient features are as follows:

1. *Adaptive Congestion Control:* ATL incorporates a new adaptive congestion control for its new reliable data transport protocol, TCP-ATL, and its new rate control protocol for multimedia traffic, RCP-ATL. The additive-increase multiplicative-decrease (AIMD) control parameters of both protocols are dynamically adjusted according to the current wireless link conditions, i.e., the packet loss rate and the access delay. This enables ATL to achieve high throughput performance despite the architectural diversity of the NGWI. The adaptive congestion control scheme described in Section 4.3 also implicitly addresses the effects of blackouts.

2. *Multimedia Support:* A new UDP-based rate control protocol, RCP-ATL, for multimedia traffic in the NGWI is also embodied within the ATL. RCP-ATL performs rate control scheme by using the adaptive congestion control to achieve high goodput performance in the heterogeneous wireless architectures. RCP-ATL also co-operates with an adaptive encoder to achieve homogeneous and smooth quality variation as explained in Section 4.4.2

3. *Fairness:* The methods for dynamic adjustment of the AIMD parameters of adaptive congestion control for both TCP-ATL and RCP-ATL are developed in Section 4.3 by

taking the fairness into consideration. Therefore, ATL protocols inherently preserve fairness to the wired TCP sources sharing the same bottleneck.

4. *Low Complexity & Backward Compatibility:* The ATL protocols are developed based on the current TCP and UDP protocols with additional functionalities tailored to address the unique requirements of the NGWI. Therefore, ATL can easily communicate with current transport layer implementations, which follow conventional TCP/IP semantics. This approach also matches with a unified, low complexity adaptive transport layer requirement for the processing and memory constrained mobile terminals.

4.2 Related Work

Despite the extensive research in transport protocols to address the challenges in wireless domain [99], [31], [20], [17], [10], [11], [57], most of them address these challenges only for a specific wireless environment and hence do not solve the problems pertaining to the NGWI. For example, independent solutions have been proposed for micro-cell [20] and macro-cell [99] environments, and for satellite networks [10], [11], [57]. The Snoop protocol [20] proposed primarily for WLANs has been shown in [99] to be inefficient in 3G environments due to its assumption of insignificant wireless link delays compared to the end-to-end delays. On the other hand, WTCP proposed in [99] for wireless wide area networks (WWAN) relies on the inter-packet separation as a congestion metric. Hence, WTCP is not applicable to WLAN and pico-cells, since its congestion metric is not reliable in the high-bandwidth low-delay environments. In [31], TCP-Westwood is proposed to improve the TCP performance in the hybrid wired/wireless networks. TCP-Westwood uses ACK reception rate to estimate the available bandwidth, which is then used to calculate the congestion window and slow start threshold. However, for the links with high delay, TCP-Westwood performance degrades due to the decrease in the estimation accuracy because of late arriving feedback [31]. Furthermore, the abrupt changes in wireless link delay due to vertical handoffs between different wireless systems of the NGWI may also decrease the accuracy of its bandwidth estimation method.

Similarly, the solutions for satellite networks also do not stand as a unified solution

to address the heterogeneities in the NGWI. TCP-Peach [10] and its enhanced version TCP-Peach+ [11] significantly improve the throughput performance in the satellite links. However, their congestion control algorithms, which are tailored to match requirements of the links with high bandwidth-delay products, may not be applicable to WLAN and pico-cell environments.

For the NGWI architecture in which the mobile user migrates between the different environments as illustrated in Figure 1, none of these protocols will suffice. In [58], an end-to-end transport layer approach called pTCP is proposed to perform bandwidth aggregation for mobile terminals that are connected to multiple wireless networks. pTCP opens and maintains one TCP-virtual connection for every interface to perform data striping. Although pTCP does not propose specific congestion control schemes to be used for each of the interfaces, it has been pointed out in [58] that incorporating interface-specific transport protocols to address the architectural heterogeneity could be a conceivable solution until a unified transport layer framework is developed. However, including each of the existing transport protocols tailored for a specific architecture in a single transport layer is not a practical solution due to the processing and memory constraints of wireless terminals. Therefore, a unified and adaptive transport protocol that adapts itself to the different wireless environments is yet to be developed to address the architectural heterogeneities posed by the NGWI objective.

On the other hand, there exists a necessity of a rate control mechanism for multimedia flows to avoid unfairness to TCP sources and further congestion collapse [52]. Although there exists a significant amount of research in this context [88], [73], [32], [87], [18], most of them are proposed for wired networks. These solutions follow the conservative rate halving behavior of TCP, which leads to unnecessary rate throttle and hence severe performance degradation in the wireless environments. In [127], an end-to-end wireless multimedia streaming TCP-Friendly protocol (WMSTFP) is proposed for multimedia over wireless Internet. In WMSTFP, the sender adjusts the sending rate by calculating the available bandwidth estimation based on the feedback obtained from the receiver. Although the WMSTFP improves the network utilization and maintain fairness to the TCP sources, it

is not developed to address all of the challenges in the heterogeneous architectures of the NGWI. For example, in the satellite networks, the RTT-based rate estimation technique used in WMSTFP may not achieve high utilization of the link resources due to very high propagation delay. In [106], Rate Control Scheme (RCS) has been proposed for real-time traffic in the networks with high bandwidth-delay products and high bit error rates. RCS significantly improves the throughput performance while maintaining fairness. However, its *dummy packet* based congestion control algorithm is specifically tailored to match the requirements of the links with high propagation delay and may be inefficient for the wireless environments with low access delay such as WLAN and pico-cells.

Despite the significant amount of research in transport protocols for wireless networks and rate control schemes for multimedia traffic flows, there is no single solution to address all of these heterogeneities pertaining to the NGWI. Hence, there exists a need for a unified transport layer to handle both architectural diversity and heterogeneous application requirements of the NGWI.

4.3 Adaptive Congestion Control

In this section, the variation in the performance degradation severity due to the architectural heterogeneity in the NGWI is first investigated via a case study. Then the adaptive congestion control for reliable data transport and multimedia traffic delivery are developed.

4.3.1 Architectural Heterogeneity: Case Study

The performance degradation experienced by current TCP protocols is primarily affected by the wireless link conditions. The increase in the probability, p_w, of packet loss in the wireless link also increases the severity of the performance degradation. Similarly, an increase in the round-trip time, RTT, of the end-to-end path due to an additional wireless link delay, d_w, further degrades the network utilization [10]. Moreover, the high RTT also prevents terminal from utilizing its fair bandwidth share because of the well-known unfairness to the TCP flows with higher RTT [72]. Therefore, the severity of the performance degradation changes according to the varying wireless link conditions.

In order to study the effects of the architectural heterogeneity in the NGWI on the TCP

Table 2: Simulation parameters and results.

Wireless System	$\mathbf{d_w}$ (ms)	$\mathbf{p_w}$ (Mb/s)	***B/W*** (Mb/s)	***Tput***	***Gput***
Wired	10	0	1	0.7286	0.86555
WLAN	10	10^{-4}	1	0.7271	0.84686
3G Cellular	50	10^{-3}	1	0.4899	0.57977
Satellite	150	10^{-2}	1	0.1549	0.12417

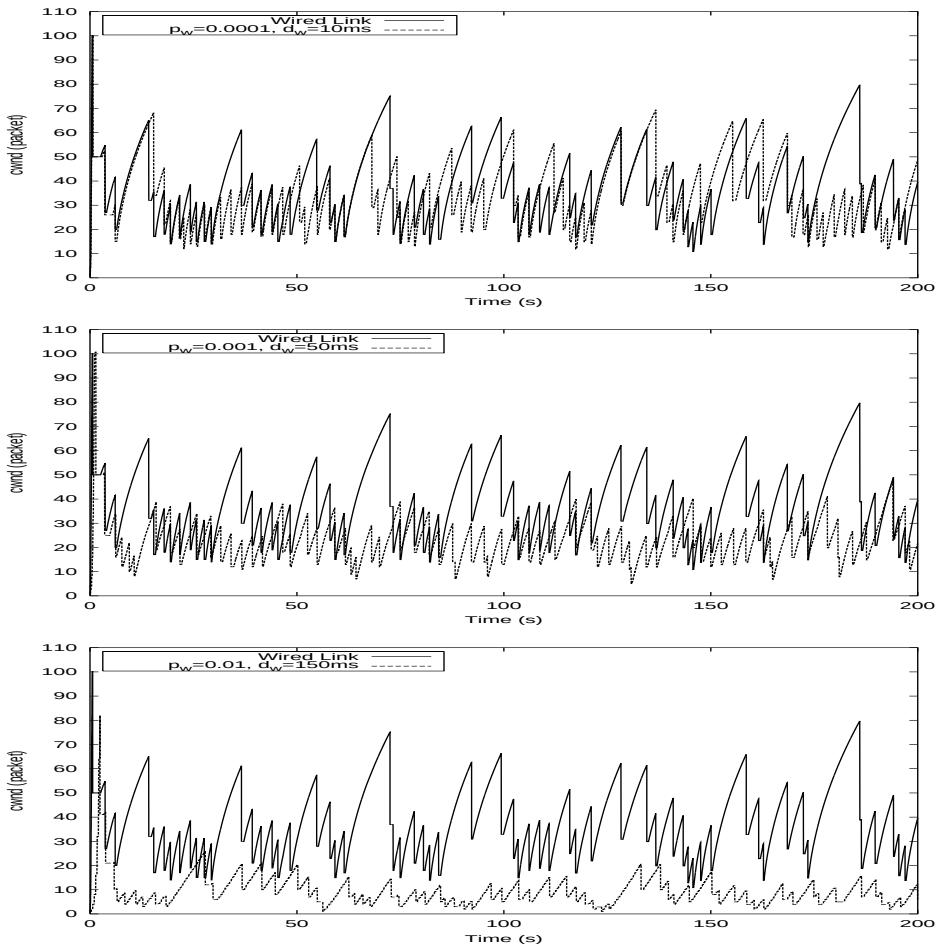

Figure 47: Congestion window change with time for different wireless link conditions.

performance, simulation experiments are performed using *ns-2* [111]. Here, 3 wireless and 1 wired TCP-Newreno sources are connected to 4 receivers via 2.5 Mb/s bottleneck link. The wireless TCP sources are connected to the bottleneck via links with the capacity of 1Mb/s, the packet loss probabilities of $p_w = 10^{-4}, 10^{-3}, 10^{-2}$ and the one-way link delays of $d_w = 10, 50, 150$ ms, respectively. All TCP sources are attached with an FTP application.

Note that three (p_w, d_w) pairs used in the simulation experiments represent the typical wireless link conditions in WLAN, 3G cellular, and satellite networks.

The congestion window (*cwnd*) change of the TCP sources are shown in Figure 47. The wired TCP source halves its congestion window only in case of packet loss due to congestion. However, the wireless TCP sources perform more rate throttles, which is proportional to the p_w as shown in Figure 47. Furthermore, the recovery from *cwnd* halving takes time proportional to the RTT, which increases with the wireless link delay d_w. While the performance of the wireless source connected to WLAN is very close to the wired TCP source, TCP source representing satellite connection almost cannot utilize its entire link as shown in Figure 47. The parameters and the results of the simulation experiments are summarized in Table 2.

It is observed in Figure 47 and Table 2 that the achievable throughput performance significantly varies with varying wireless environment conditions. The same experiments are also conducted with an AIMD rate control protocol [88] for multimedia traffic and similar patterns have been observed and their goodput results are also given in Table 2. These results motivate us to develop a unified and adaptive transport layer protocols that maintain the maximum achievable throughput in all wireless architectures of the NGWI.

4.3.2 Adaptive Congestion Control for Reliable Data Transport

It is shown in Section 4.3.1 that the TCP performance significantly varies with wireless link conditions. Therefore a new adaptive congestion control is necessary to maintain the highest throughput performance possible in all wireless architectures. On the other hand, this objective is upper bounded with the throughput of the wired TCP source sharing the same bottleneck in order to maintain the fairness. For example, the *cwnd* curve achieved by the wired TCP in Figure 47 represents the target throughput to achieve for the other three wireless resources. Therefore, the objective for adaptive congestion control can be restated as the minimization of the difference between the areas under the *cwnd* curves achieved by the wired TCP and the wireless TCP sources.

As observed in Figure 47, the TCP throughput directly depends on the link conditions. Let $\mathcal{T}_{\alpha,\beta}(p, R, T_0, b)$ be the throughput achieved by a TCP source, where (α,β) are its additive-increase and multiplicative-decrease parameters; p is the packet loss probability; R is the end-to-end round-trip time (RTT); T_0 is the initial retransmission timeout (RTO), and b is the number of data packets acknowledged with a single ACK. Hence, the wired TCP throughput in Figure 47 can be expressed by $\mathcal{T}_{1,\frac{1}{2}}(p_c, R_c, T_{0c}, b)$, where p_c is the probability of packet loss experienced due to congestion, and R_c is the end-to-end RTT of the connection without any additional wireless link delay. Therefore, $\mathcal{T}_{1,\frac{1}{2}}(p_c, R_c, T_{0c}, b)$ is the upper bound for the target throughput due to fairness consideration. Hence, the objective of the adaptive congestion control method is to achieve $\mathcal{T}_{1,\frac{1}{2}}(p_c, R_c, T_{0c}, b)$ regardless of the underlying wireless architecture and the link conditions.

In the simulation experiments presented in Section 4.3.1, the wireless TCP sources achieved lower throughput than the wired TCP source due to the increased packet loss rate and RTT because of the additional packet losses and the link delay experienced in the wireless link. This situation can be generalized for $p > p_c$ and $R > R_c$ as

$$\mathcal{T}_{1,\frac{1}{2}}(p, R, T_0, b) \quad < \quad \mathcal{T}_{1,\frac{1}{2}}(p_c, R_c, T_{0c}, b) \tag{37}$$

where p and R are the total probability of packet loss due to both wireless link and congestion and the end-to-end round-trip time (RTT), respectively. Here, p and R can be expressed by

$$p \quad = \quad 1 - (1 - p_w)(1 - p_c) \tag{38}$$

$$R \quad = \quad R_c + 2d_w \tag{39}$$

where p_w is the probability of wireless link related packet loss; p_c is the probability of packet loss due to congestion; R_c is the end-to-end RTT of the connection without any additional wireless link delay; and d_w is the one-way wireless link delay as in Table 2.

As the mobile terminal migrates between different wireless environments of the NGWI; p_w and d_w, and hence p and R in (38) and (39) vary continuously. Such variation in the link conditions leads the mobile terminal to experience different throughput degradation levels as observed in Figure 47.

To maintain the highest throughput performance in all of the different wireless architectures characterized by different access delays and link error rates, and achieve fairness to the wired TCP sources sharing the same bottleneck; ATL incorporates a new adaptive congestion control that dynamically adjusts its AIMD parameters (α,β) according to the current wireless link conditions, i.e., p_w and d_w. Therefore, the objective of the new adaptive congestion control is to obtain (α, β) pair such that

$$\mathcal{T}_{\alpha,\beta}(p, R, T_0, b) \approx \mathcal{T}_{1,\frac{1}{2}}(p_c, R_c, T_{0c}, b) \tag{40}$$

is achieved in all of the different architectures of the NGWI; i.e., $\forall p_w$ and $\forall d_w$ for $p > p_c$ and $R > R_c$, where p and R are expressed in (38) and (39) as a function of wireless link condition parameters p_w and d_w.

Let $\widehat{T}$ be the upper bound for the target throughput to be achieved by the wireless TCP sources, i.e., $\widehat{T} = \mathcal{T}_{1,\frac{1}{2}}(p_c, R_c, T_{0c}, b)$. Recall that this upper bound is imposed by the fairness consideration and hence $\widehat{T}$ is the throughput achieved by the wired TCP source experiencing p_c and R_c as the packet loss probability due to congestion and the end-to-end RTT (with no additional wireless link delay), respectively. Given that p, R, p_w, and d_w are known, it follows from (38) and (39) that p_c and R_c can be calculated as

$$p_c = \frac{p - p_w}{1 - p_w} \tag{41}$$

$$R_c = R - 2d_w \tag{42}$$

The throughput of the TCP connection $\mathcal{T}_{\alpha,\beta}(p, R, T_0, b)$ in (40) can be expressed by [126]

$$\mathcal{T}_{\alpha,\beta}(p, R, T_0, b) = \frac{1}{R\sqrt{\frac{2b(1-\beta)p}{\alpha(1+\beta)}} + T_0\left(3\sqrt{\frac{(1-\beta^2)bp}{2\alpha}}\right)p(1+32p^2)} \tag{43}$$

where (α,β) are the additive-increase and multiplicative-decrease parameters; p is the packet loss probability; R is the end-to-end round-trip time (RTT); T_0 is the initial retransmission timeout (RTO), and b is the number of data packets acknowledged with a single ACK. Therefore, $\widehat{T}$ can be calculated by feeding (43) with p_c, R_c, T_{0c}, and b. Here, each data packet can be assumed to be acknowledged by an ACK, i.e., $b = 1$, and T_{0c} can be approximately set to $4R_c$ to provide fairness with TCP [52]. Therefore, by substituting the obtained $\widehat{T}$

and (43) into (40), it is obtained that

$$R\sqrt{\frac{2b(1-\beta)p}{\alpha(1+\beta)}} + T_0\left(3\sqrt{\frac{(1-\beta^2)bp}{2\alpha}}\right)p(1+32p^2) = \frac{1}{\widehat{T}} \tag{44}$$

Consequently, the additive-increase parameter α that achieves the objective stated in (40) can be calculated from (44) as

$$\alpha = \frac{bp(1-\beta)}{2(1+\beta)}\left[\widehat{T}\left(2R + 3T_0p(1+32p^2)(1+\beta)\right)\right]^2 \tag{45}$$

Therefore, once p_w, d_w, p, and R are known, $\widehat{T}$ can be calculated by (43); then (45) can be used to set the additive-increase parameter α for $\forall\beta \in [0.5, 1)$. The selection of the multiplicative-decrease factor β from $[0.5, 1)$ is discussed in Section 4.4.1 and Section 4.5.1.1. By this way, the throughput degradation can be avoided and highest throughput, which is upper bounded by the fairness constraint, is achieved via dynamically adjustment of AIMD parameters according to the wireless link conditions. The detailed operation of the adaptive congestion control for reliable data transport is presented in Section 4.4.1.

4.3.3 Adaptive Rate Control for Multimedia Traffic

Similarly, a new adaptive rate control for multimedia flows is necessary to maintain the highest performance possible in all of the different wireless architectures while preserving fairness to the TCP sources. As in Section 4.3.2, the highest throughput is again upper bounded by the throughput of the wired TCP source sharing the same bottleneck in order to maintain fairness.

Assume a general *rate-based* AIMD scheme which increases the transmission rate, S, additively with α at each RTT, i.e., $S = S + \alpha$; and reduces it multiplicatively by β if a packet loss is detected, i.e., $S = S \cdot \beta$. The steady-state throughput of such *rate-based* AIMD scheme can be expressed by [8]

$$\mathcal{T}^r_{\alpha,\beta}(p, R) = \frac{\alpha}{4(1-\beta)}\left[1 + \beta + \sqrt{(3-\beta)^2 + \frac{8(1-\beta^2)}{\alpha Rp}}\right] \tag{46}$$

where p is the packet loss probability; R is the RTT; and α and β are the additive-increase and the multiplicative-decrease factors, respectively.

It is observed from Table 2 and (46) that the throughput of the rate-based AIMD scheme depends on the link conditions, i.e., p and R, and the values of α and β. In order to maintain the highest throughput performance and achieve fairness to the wired TCP sources, ATL adapts the AIMD parameters of its rate control protocol to the link conditions, i.e., p_w and d_w. As in Section 4.3.2, the objective of the new adaptive rate control is to obtain (α, β) pair such that

$$\mathcal{T}^r_{\alpha,\beta}(p, R) \approx \mathcal{T}_{1,\frac{1}{2}}(p_c, R_c, T_{0c}, b) \tag{47}$$

is achieved in all of the different architectures of the NGWI; i.e., $\forall p_w$ and $\forall d_w$ for $p > p_c$ and $R > R_c$, where p and R are expressed in (38) and (39) as a function of wireless link condition parameters p_w and d_w.

Let $\widehat{T}$ be the upper bound for the target throughput to be achieved by the wireless multimedia traffic sources as in Section 4.3.2, i.e., $\widehat{T} = \mathcal{T}_{1,\frac{1}{2}}(p_c, R_c, T_{0c}, b)$. Thus, by substituting the $\widehat{T}$ and (46) into (47)

$$\frac{\alpha}{4(1-\beta)}\left[1+\beta+\sqrt{(3-\beta)^2+\frac{8(1-\beta^2)}{\alpha R p}}\right] = \widehat{T} \tag{48}$$

Hence, the additive-increase parameter α of the AIMD-based rate control protocol to achieve a target throughput of $\widehat{T}$ can be obtained as

$$\alpha = \frac{(1+\beta)}{2}\left(\widehat{T}+\frac{1}{R\cdot p}\right)\left[\sqrt{1+\frac{8\widehat{T}^2(1-\beta)}{\left(\widehat{T}+\frac{1}{R\cdot p}\right)^2(1+\beta)^2}}-1\right] \tag{49}$$

Therefore, given that p_w, d_w, p, and R are known, then (49) can be used to set α for $\forall\beta \in [0.5, 1)$ according to the wireless link conditions. The selection of the multiplicative-decrease factor β from $[0.5, 1)$ is discussed in Section 4.4.1 and Section 4.5.1.1. By this way, an adaptive rate control scheme can achieve the highest possible throughput while maintaining fairness. The detailed operation of the adaptive rate control scheme for multimedia traffic is explained in Section 4.4.2.

4.4 ATL: Adaptive Transport Layer for NGWI

ATL is a unified adaptive transport layer that incorporates a new reliable data transport protocol, TCP-ATL, and a new rate control protocol, RCP-ATL, for multimedia traffic.

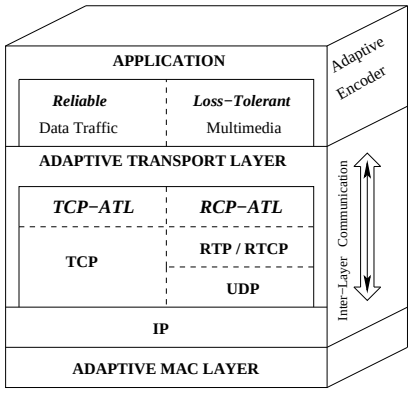

Figure 48: A typical protocol stack including Adaptive Transport Layer (ATL).

The protocol structure of ATL is illustrated in Figure 48. According to the service type requested by the application layer or the incoming request from the receiver side, the appropriate protocol algorithm is initiated. The application layer interfacing and the connection establishment procedures are left out of the scope of this work. In this section, the operations of the TCP-ATL and the RCP-ATL are presented in detail.

4.4.1 TCP-ATL: Adaptive Reliable Data Transport Protocol

ATL incorporates a new adaptive reliable transport protocol, TCP-ATL, to address the architectural heterogeneity of the NGWI. TCP-ATL inherits well-established TCP functionalities of the legacy TCP protocols [104] and utilizes the adaptive congestion control scheme developed in Section 4.3.2. TCP-ATL can be also implemented based on any conventional AIMD TCP protocol to achieve the adaptive congestion control objective. Furthermore, the selective acknowledgment (SACK) options [81] are shown to be very effective in recovering multiple packet losses in one TCP window. This is much more important on the links with high bandwidth-delay product such as satellite links. Therefore, TCP-ATL uses the selective acknowledgment (SACK) options [81] to guarantee reliability.

In Section 4.3.2, it is shown that the AIMD parameters of TCP can be adjusted according to the link conditions by using (45) given that p, R, p_w, and d_w are known. To utilize (45) for this goal, the TCP-ATL source continuously measures the packet loss rate p within sliding time window of τ. The wireless link delay d_w and probability of wireless related packet loss p_w are assumed to be obtained from the underlying *adaptive MAC layer*, which is also

essential to address the architectural heterogeneities in the NGWI. In practice, the wireless MAC protocols have the information about the packet loss rate and the delay experienced in the wireless access link, i.e., p_w and d_w. The access delay is immediately available at the MAC layer, since many MAC protocols use it to calculate their timeout values, e.g., clear-to-send (CTS) timeout value calculation in IEEE 802.11 [59]. Furthermore, wireless MAC protocols perform several attempts to get the packets it receives from upper layer to the access point. If the MAC layer cannot successfully get the data packet across, it informs the upper layer about this result. Note that this failure may occur because of several wireless link conditions, such as access contention, link errors, link outage etc. Hence, the packet loss rate due to wireless link information, p_w, can be obtained from the MAC layer information via simple cross-layer interaction. The details of such communication between different networking layers are software implementation issues and left beyond the scope of this work. Moreover, note that p_w is also measured within a sliding time window of τ and includes all wireless link related packet losses encountered by the underlying MAC layer due to access failure, bit errors, fading, and signal loss due to handoff or blackout.

Once p, R, p_w, and d_w are known, p_c and R_c can be obtained by using (41) and (42). Hence, $\widehat{T} = \mathcal{T}_{1,\frac{1}{2}}(p_c, R_c, T_{0c}, b)$ can be calculated using (43) and plugged into (45) to determine α for a given β for $\forall\beta \in [0.5, 1)$. While the selection of β affects the instantaneous TCP-friendliness [49] of the TCP-ATL behavior, it does not affect the overall fairness. This is because any (α, β) pair determined by (45) inherently complies with fairness requirement, since (45) itself is obtained by taking that requirement into consideration. Therefore, β can be safely selected from $[0.5, 1)$. On the other hand, the higher the β is selected, the higher performance can be observed in the links with high bandwidth-delay products. This is mainly because the throughput degradation due to frequent rate throttles is amplified by the high bandwidth-delay product of the link. Hence, selecting higher β values from $[0.5, 1)$ help compensate the adverse effects of high propagation delay on the performance. During the simulation experiments presented in Section 4.5, β values of 0.75, 0.80, and 0.85 are found to give the best performance, for $d_w = 10$, 50, and 150 ms, respectively. Therefore, β can be set to be 0.75, 0.80, and 0.85, for WLAN (0 ms $< d_w <$ 50 ms), 3G cellular (50

ms $\leq d_w <$ 150 ms), and satellite networks ($d_w \geq$ 150 ms), respectively.

In the beginning of the connection, there is no information in terms of p, and p_w. Therefore, the connection starts with default AIMD parameters, i.e., $(\alpha, \beta) = (1, 0.5)$. Note that in the absence of a wireless link, i.e., $p_w = 0$ and $d_w = 0$, the connection simply uses normal TCP which is also desirable to maintain fairness. As packet losses are being experienced due to both congestion and wireless link, (α, β) is adjusted to adapt the protocol configuration to the varying wireless link conditions. Furthermore, at each vertical handoff, it is essential that ATL updates its AIMD parameters to address the possible abrupt variation in the wireless link conditions. It is assumed that the vertical handoff events are informed by underlying MAC layer over inter-layer communication plane shown in Figure 48.

On the other hand, the blackout situations are also implicitly addressed by the adaptive congestion control algorithm of TCP-ATL. If a blackout occurs due to mobility or fading by the environmental obscurations, this immediately increases p_w. The TCP-ATL source then calculates new α using (45) to adapt the congestion control configuration to the new link conditions. It follows from (45) that an increase in p_w also increases α. Therefore, TCP-ATL can quickly recover from the throughput degradation due to a blackout by increasing its congestion window more aggressively. Once the signal is back, p_w immediately drops which also reduces α. Therefore, TCP-ATL can dynamically return to its configuration just before the blackout.

The default TCP-ATL operation explained above directly applies to the scenarios, where the source is wireless and the receiver is wired terminals. The TCP-ATL operation may require slight modifications in terms of calculations of p_c and R_c according to the different source/receiver combinations. The operation of the protocol for each source/receiver combination is summarized as follows:

1. **Wireless Source/Wired Receiver**: In this case, the default ATL operation explained in this section does not require any modifications.

2. **Wired Source/Wireless Receiver**: In this case, the wireless link parameters, i.e.,

p_w and d_w, are supplied to the source. Once the wireless terminal acquires p_w and d_w for the first time, it forwards them to the source within data ACK packets to avoid any additional overhead. p_w and d_w are then forwarded to the source if they vary. Similarly, the vertical handoff event is also signaled to the source using data ACK packets in this scenario.

3. **Wireless Source/Wireless Receiver**: In this case, the end-to-end path includes two wireless links. Let p_w^s, d_w^s, and p_w^r, d_w^r be the packet loss probability and the delay of the wireless link at the source and the receiver side of the end-to-end path, respectively. As the wireless receiver first obtains p_w^r and d_w^r, it forwards them to the source as explained above. The wireless ATL source, which receives this information, then infers that the receiver is also wireless terminal equipped with the ATL. In this case, the total packet loss probability p and the round-trip time R, which are previously expressed by (38) and (39), can be calculated as

$$p = 1 - (1 - p_w^s)(1 - p_c)(1 - p_w^r) \tag{50}$$

$$R = R_c + 2(d_w^s + d_w^r) \tag{51}$$

It follows from (50) and (51) that the wireless source, which measures p and R as explained before, can calculate p_c and R_c as

$$p_c = 1 - \frac{1 - p}{(1 - p_w^s)(1 - p_w^r)} \tag{52}$$

$$R_c = R - 2(d_w^s + d_w^r) \tag{53}$$

Once p_c and R_c are found using (52) and (53), the original TCP-ATL operation is then resumed.

4. **Wired Source/Wired Receiver**: In this case, no wireless link involves in the entire connection path, that is, $p_w = 0$ and $d_w = 0$. Therefore, normal TCP protocol operation inherited by TCP-ATL resumes the connection. Note that ATL is also backward compatible as it can still establish TCP connection even if any end-party is not equipped with the ATL.

The operation of the adaptive congestion control scheme used by TCP-ATL protocol is summarized in the pseudo-algorithm shown in Figure 49.

Configure_AIMD()
  /* Case 1 */
  if (Wireless Source / Wired Receiver)
    Obtain p_w and d_w from MAC layer;
    $p_c = \frac{p - p_w}{1 - p_w}$;
    $R_c = R - 2d_w$;
  end;
  /* Case 2 */
  if (Wired Source / Wireless Receiver)
    Obtain p_w and d_w from RECEIVER;
    $p_c = \frac{p - p_w}{1 - p_w}$;
    $R_c = R - 2d_w$;
  end;
  /* Case 3 */
  if (Wireless Source / Receiver)
    Obtain p_w^s and d_w^s from MAC layer;
    Obtain p_w^r and d_w^r from RECEIVER;
    $d_w = d_w^s + d_w^r$;
    $p_c = 1 - \frac{1-p}{(1-p_w^s)(1-p_w^r)}$;
    $R_c = R - 2(d_w^s + d_w^r)$;
  end;
  Calculate $\widehat{T} = \mathcal{T}_{1,\frac{1}{2}}(p_c, R_c, T_{0c}, b)$ with (43);
  if (0 ms $< d_w <$ 50 ms)
    Set $\beta = 0.75$;
  if (50 ms $\leq d_w <$ 150 ms)
    Set $\beta = 0.80$;
  if ($d_w \geq$ 150 ms)
    Set $\beta = 0.85$;
  if (TCP_ATL)
    Set α with (45);
  else if (RCP_ATL)
    Set α with (49);
  end;
end;

Figure 49: Pseudo-algorithm for the operation of the adaptive congestion and rate control schemes used by TCP-ATL and RCP-ATL protocols.

4.4.2 RCP-ATL: Adaptive Rate Control Protocol for Multimedia Traffic

ATL incorporates a new adaptive rate control protocol, RCP-ATL, to address the architectural heterogeneity of the NGWI. RCP-ATL is an end-to-end rate control protocol that uses the adaptive rate control scheme developed in Section 4.3.3 in order to produce TCP-friendly traffic flows while maintaining high throughput performance in the NGWI.

RCP-ATL can run on top of RTP/RTCP [94] and UDP as shown in Figure 48. RCP-ATL

is not an ARQ protocol and the source does not perform retransmission due to tighter time constraints of the multimedia flows. However, the receiver sends back an acknowledgment (ACK) for any received packet. If the RTP/RTCP is implemented, then these ACKs can be the receiver reports (RR) that includes the reception quality statistics such as the number of packets received, RTP timestamp, fraction lost, and cumulative number of packets lost. Hence, RCP-ATL can obtain p and R from the RTP receiver reports. If RTP/RTCP is not used for any reason, then data ACKs would be sufficient to acquire p and R.

RCP-ATL follows the AIMD rate control behavior. It multiplicatively decreases its data rate (S) in case of a packet loss, i.e., $S = S \cdot \beta$. Otherwise, it additively increases the data rate with α at each RTT, i.e., $S = S + \alpha$. Furthermore, RCP-ATL can be implemented based on any existing AIMD rate control protocol to achieve the adaptive congestion control objective.

As in Section 4.4.1, p_w and d_w are assumed to be obtained from the underlying *adaptive MAC layer*. Once p, R, p_w, and d_w are known, p_c and R_c can be obtained by using (41) and (42). Hence, $\widehat{T} = \mathcal{T}_{1,\frac{1}{2}}(p_c, R_c, T_{0c}, b)$ can be calculated using (43) and plugged into (49) to determine α for a given β for $\forall\beta \in [0.5, 1)$. The discussions on the selection of β given in Section 4.4.1 also apply here. In this case, the selection of β also affects the smoothness of the rate variation, which is essential to maintain a certain quality level in multimedia streaming. Although any (α, β) pair determined by (49) inherently complies with fairness requirement, higher β is preferable for smoothness requirement. The investigation of the effects of the β selection on the smoothness of the rate variation is left for future study.

On the other hand, RCP-ATL rate control protocol can be in co-operation with an adaptive media encoding process. The adaptive encoder can be provided with the available bandwidth $S(t)$ estimated by the RCP-ATL. Hence, the real-time multimedia encoder can then adapt its encoding rate $R(t)$ according to the controlled fair share of the network resources, i.e., $R(t) \approx S(t)$. By this way, the quality of the encoded multimedia can vary homogeneously and smoothly without leading to congestion and undesirable abrupt quality variations.

As explained for TCP-ATL in Section 4.4.1, RCP-ATL also indirectly addresses the

blackout situations by its adaptive rate control algorithm. On the other hand, the RCP-ATL may require simple modifications in terms of calculations of p_c and R_c according to the different source/receiver combinations. The explanation of the TCP-ATL operation given in Section 4.4.1 for each source/receiver combination also applies to RCP-ATL and hence is not repeated here. The only difference is that p_w, d_w and vertical handoff event information can now be forwarded to the source within RTP receiver reports rather than data ACKs. However, if RTP/RTCP is not used for any reason, p_w, d_w, and vertical handoff event information can be still sent to the source within data ACKs. The operation of the adaptive rate control scheme used by RCP-ATL protocol is also summarized in the pseudo-algorithm shown in Figure 49.

4.5 Performance Evaluation

In order to investigate the performance of the ATL, extensive simulation experiments are conducted. The throughput, blackouts, and fairness performance of TCP-ATL in the NGWI are evaluated in Section 4.5.1. In Section 4.5.2, the performance of RCP-ATL in terms of throughput, fairness, and jitter is investigated.

4.5.1 TCP-ATL Performance

4.5.1.1 Throughput

TCP-ATL simulation experiments are performed for varying packet loss probability (p_w) between 10^{-6} and 10^{-1}. The experiments are conducted for three different one-way wireless link delay values, i.e., $d_w = 10, 50, 150$ ms, which represent typical wireless link delay values in WLAN, 3G cellular and satellite networks, respectively. During these three sets of experiments, the multiplicative-decrease factor β is set to be 0.75, 0.80, and 0.85, respectively. These settings are found to give the best performance for simulation experiments. This makes perfectly sense since the throughput degradation (due to frequent rate throttles) increases with the bandwidth-delay product of the link. Therefore, the higher the β is selected, the higher performance is observed in the links with high bandwidth-delay products. The throughput values are obtained at the sender. The simulations are run on the following network configuration. The source and the destinations are connected to each

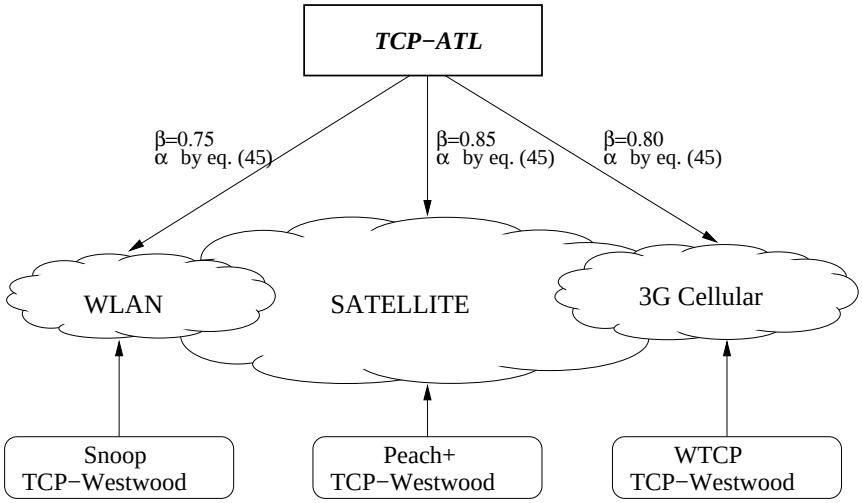

Figure 50: A unified and adaptive TCP-ATL protocol instead of using different transport protocols that are specifically designed for different wireless architectures.

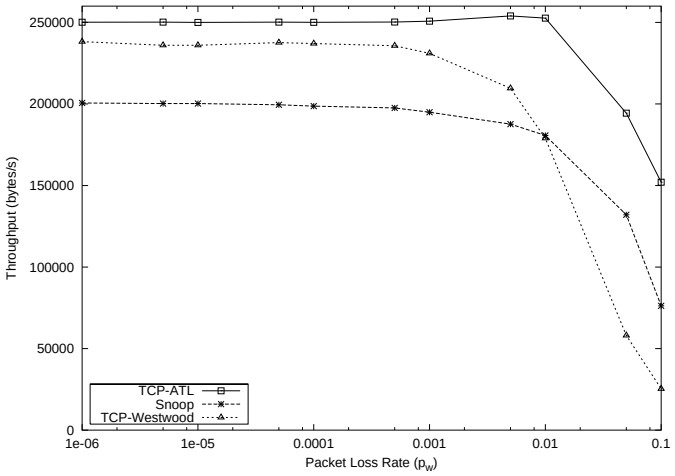

Figure 51: Throughput comparison of TCP-ATL, Snoop, and TCP-Westwood, for $d_w = 10$ ms and varying p_w.

other via two routers connected through the shared wired bottleneck. The wireless nodes are connected to the ingress router of the wired shared bottleneck via the wireless access links of these heterogeneous wireless architectures. The wired nodes are connected to the same bottleneck via wired links. The link capacities and data packet size are assumed to be 2Mb/s and 1KB, respectively. The one-way delay of the wired bottleneck link is assumed to be 10 ms. Each data packet is assumed to be acknowledged by a single ACK, i.e., $b = 1$. The packets are lost in the wireless link with the packet loss probability of p_w. The sliding window duration for the measurement of p_w is set to be $10 \cdot RTT$. The

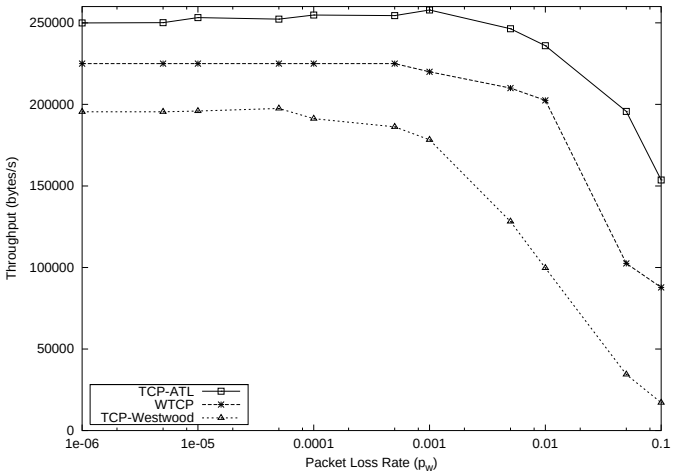

Figure 52: Throughput comparison of TCP-ATL, WTCP, and TCP-Westwood for $d_w = 50$ ms and varying p_w.

investigation of the effects of these parameters on the protocol performance are left for future study. All simulations are performed for a duration of 600 seconds. For low delay environments such as WLAN, Snoop [20] is proven to be an efficient TCP-aware link layer solution that significantly increases the TCP performance. WTCP [99] is also known to achieve the highest throughput performance in WWAN environments. In the literature, TCP-Westwood [31] is shown to significantly improve the throughput performance in the hybrid wireless/wired networks. Similarly, TCP-Peach+ [10], [11] is shown to be the best reliable transport protocol for satellite networks. Therefore, in order to show that TCP-ATL attains very high performance in all of the heterogeneous wireless environments as shown in Figure 50, its performance is compared to the performance of Snoop (with TCP-Sack) [20] and TCP-Westwood [31] for WLAN environments; WTCP [99] and TCP-Westwood [31] for wide-area 3G cellular, TCP-Peach+ [10], [11] and TCP-Westwood [31] for satellite environments.

- **WLAN:** The results of three set of experiments for $d_w = 10, 50, 150$ ms and varying p_w are plotted in Figure 51, 52, and 53, respectively. As shown in Figure 51, both TCP-ATL and TCP-Westwood outperforms Snoop for low packet loss rates, i.e., $p_w < 0.005$ and $d_w = 10$ ms. As p_w increases, $p_w \geq 0.005$, TCP-Westwood experiences much more significant performance degradation compared to TCP-ATL and Snoop

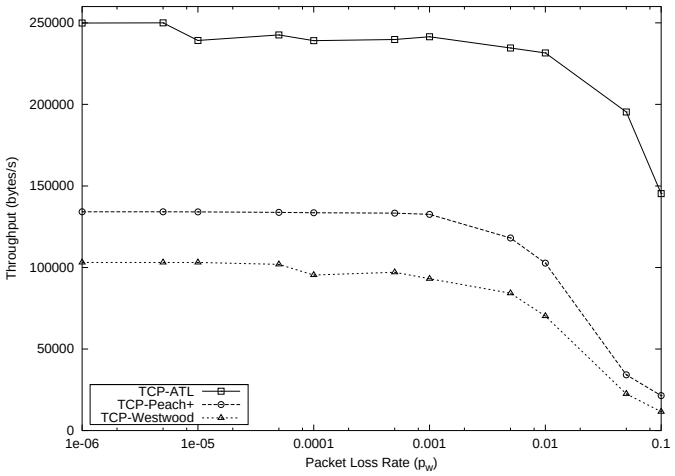

Figure 53: Throughput comparison of TCP-ATL, TCP-Westwood, and TCP-Peach+ for $d_w = 150$ ms and varying p_w.

(with TCP-Sack). This is because the adaptive congestion control of TCP-ATL helps to compensate the throughput degradation by dynamically adjusting its AIMD parameters as explained in Section 4.4.1. Snoop also achieves high performance even in such high packet loss rates because of its local retransmission strategy. Nevertheless, it does not compensate for the effects of the additional wireless access delay on the throughput efficiency. Moreover, in the Snoop implementation, the additional buffering performed at the access point in order to snoop the end-to-end connection brings additional increase in the access delay. This further amplifies the throughput performance difference between the Snoop and TCP-ATL which can handle the adverse effects of the wireless access delay as explained in Section 4.3.2. For a typical WLAN environment with a wide range of p_w, TCP-ATL significantly improves throughput over Snoop and TCP-Westwood. For $p_w = 5.10^{-2}$, TCP-ATL outperforms TCP-Westwood approximately by 234% and Snoop by 47%.

- **3G Cellular:** For this environment, the performance of TCP-ATL is also compared to WTCP in addition to TCP-Westwood. As shown in Figure 52, TCP-ATL outperforms WTCP for $d_w = 50$ ms and for all p_w values from 10^{-6} to 10^{-1}. However, WTCP achieves higher throughput performance than TCP-Westwood. As the wireless link delay increases, i.e., $d_w = 50$ ms, the throughput improvement by TCP-ATL

over TCP-Westwood further increases as shown in Figure 52. This is mainly because TCP-ATL performance does not degrade with increasing d_w while TCP-Westwood performance is significantly affected. This is because the bandwidth estimation based on the ACK reception rate used by the TCP-Westwood starts to lose accuracy because of late arriving feedback [31]. Furthermore, WTCP also outperforms TCP-Westwood in this environment due to the same reason. For $p_w = 10^{-3}$, TCP-Westwood experiences 28% throughput degradation due to the increase in d_w from 10 ms to 50 ms while TCP-ATL performance almost remains unaffected. For $p_w = 5.10^{-2}$ and $d_w = 50$ ms, TCP-ATL outperforms WTCP with 91% and TCP-Westwood with approximately 468% throughput improvement.

- **Satellite:** As shown in Figure 53, TCP-ATL throughput does not significantly degrade in the typical satellite environment, i.e., $d_w = 150$ ms, compared to the previous scenarios. Recall that $d_w = 150$ ms is the one-way wireless link delay. For $p_w = 10^{-3}$ and $d_w = 150$ ms, the TCP-ATL performance decreases by only 6% compared to $d_w = 50$ ms while TCP-Westwood experiences 48% throughput degradation. This is again because of the adaptive congestion control scheme of the TCP-ATL, which updates its AIMD parameters according to the p_w and d_w. TCP-ATL throughput also shows similar behavior for varying p_w as shown in Figure 53. However, the throughput degradation due to high p_w is not as severe as it is for TCP-Westwood and TCP-Peach+. TCP-ATL improves the throughput up to 7.6 times over TCP-Westwood for $d_w = 150$ ms. Note also that TCP-ATL outperforms TCP-Peach+ by 125% throughput improvement for $p_w = 10^{-2}$ and $d_w = 150$ ms.

Consequently, instead of using different transport protocols developed for specific architectures, TCP-ATL achieves high throughput performance in all of the three different wireless architectures by adapting its protocol configuration as shown in Figure 50.

4.5.1.2 Roaming Between Heterogeneous Architectures

Simulation experiments are also performed to evaluate the throughput achieved by the TCP-ATL while the mobile terminal roams between different wireless architectures. Here,

Table 3: Throughput achieved while (wireless source) roaming between heterogeneous wireless architectures.

Exp. #	$(\mathbf{t_1}, \mathbf{t_2})$	(N_1, N_2, N_3)	***Tput (Mb/s)***
1	(100,300)	(WLAN,3G,Sat)	1.8870
2	(200,300)	(WLAN,Sat,3G)	1.9149
3	(200,500)	(Sat,WLAN,3G)	1.8919
4	(200,400)	(Sat,3G,WLAN)	1.8979
5	(100,200)	(3G,WLAN,Sat)	1.8701
6	(400,500)	(3G,Sat,WLAN)	1.9210

the mobile terminal performs vertical handoff between three different wireless architectures (WLAN, 3G, Satellite), i.e., N_1, N_2, and N_3, at $t = t_1$ and $t = t_2$ during its connection period. The simulations are performed for a duration of 600 seconds. $p_w = 10^{-5}, 10^{-4}, 10^{-3}$ and $d_w = 10, 50, 150$ ms for WLAN, 3G cellular, and satellite environments, respectively. The other simulation parameters are the same as the ones used in Section 4.5.1.1. The simulations are performed for various (t_1, t_2) and (N_1, N_2, N_3) combinations. Furthermore, the same simulation experiments are also performed for two different cases: (i) the wireless mobile terminal is the source and (ii) the wireless mobile terminal is the receiver.

- **Wireless Source:** In this case, the wireless mobile terminal is the source and roaming between heterogeneous wireless architectures of the NGWI while it has an active connection. For example, in the first experiment, the connection starts in WLAN at $t = 0$, and then at $t_1 = 100$ seconds the source migrates from WLAN to 3G and at $t_2 = 300$ seconds from 3G to the satellite environment. As shown in Table 3, TCP-ATL maintains its high throughput performance (around 1.9 Mb/s) regardless of what portion of the connection takes place in which wireless architecture.

- **Wireless Receiver:** In this scenario, where the wireless mobile terminal is the receiver, the throughput achieved by the TCP-ATL does not vary significantly as shown in Table 4. Although there exists very small throughput degradation, this is mainly due to loss of the packets carrying p_w and d_w from the wireless receiver node to the sender. In this case, TCP-ATL source cannot immediately respond to the variation in the p_w and d_w and hence throughput performance is affected. Furthermore, another

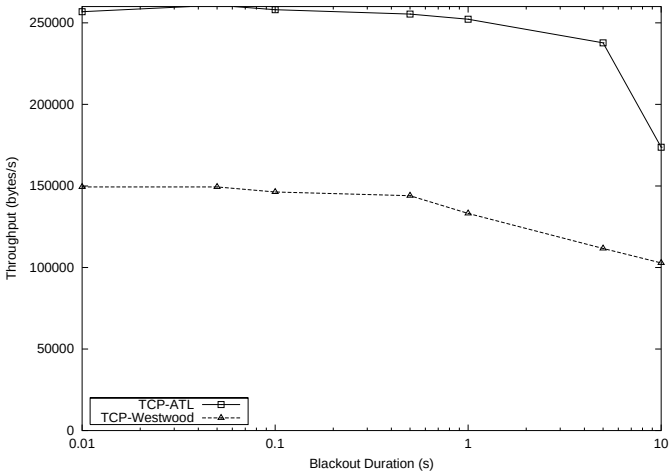

Figure 54: Throughput comparison of TCP-ATL and TCP-Westwood for varying blackout durations for $d_w = 50$ ms and $p_w = 10^{-3}$.

Table 4: Throughput achieved while (wireless receiver) roaming between heterogeneous wireless architectures.

Exp. #	$(\mathbf{t_1}, \mathbf{t_2})$	(N_1, N_2, N_3)	***Tput (Mb/s)***
1	(100,300)	(WLAN,3G,Sat)	1.8521
2	(200,300)	(WLAN,Sat,3G)	1.8943
3	(200,500)	(Sat,WLAN,3G)	1.8755
4	(200,400)	(Sat,3G,WLAN)	1.8771
5	(100,200)	(3G,WLAN,Sat)	1.8561
6	(400,500)	(3G,Sat,WLAN)	1.9093

reason for this throughput degradation is slightly delayed congestion control adaptation caused by the additional delay incurred while the wireless receiver forwards the wireless link parameters, i.e., p_w and d_w, to the source as discussed in *case 2* of the detailed protocol operation presented in Section 4.4.1. Despite such a slight variation, the achieved throughput is still around 1.9 Mb/s for all of the experiment scenarios as shown in Table 4.

Therefore, regardless of whether the wireless mobile terminal is the source or the receiver end of the end-to-end connection, TCP-ATL can provide seamless high network utilization while mobile users roam between heterogeneous wireless architectures as shown in Figure 50.

4.5.1.3 Blackout

The throughput achieved by TCP-ATL and TCP-Westwood for varying blackout durations between 10 ms and 10 seconds are given in Figure 54. Here, it is assumed that $d_w = 50$ ms and $p_w = 10^{-3}$. The simulations are performed for a duration of 600 seconds, where the blackout occurs at $t = 300$ seconds.

As shown in Figure 54, the throughput decreases with increasing blackout duration as expected. This decrease is observed in both of the curves representing the TCP-ATL and TCP-Westwood. TCP-ATL outperforms TCP-Westwood for all blackout duration values. This is achieved by the dynamically adaptive congestion control used by TCP-ATL. When a blackout occurs due to mobility or fading by environmental obscurations, this immediately increases p_w. The TCP-ATL source then calculates new α using (45) to adapt the congestion control configuration to the new link conditions. Therefore, TCP-ATL can quickly recover from the throughput degradation due to a blackout as explained in Section 4.4.1. For the blackout duration of 1 second, TCP-ATL achieves 89% throughput improvement over TCP-Westwood.

4.5.1.4 Fairness

For the fairness performance of TCP-ATL, two different scenarios, i.e., Intra-protocol fairness and Fairness to wired TCP flows, are considered. For the former case, the fairness among TCP-ATL flows sharing the same bottleneck is investigated. For the latter case, the fairness of TCP-ATL to the wired TCP sources, which are sharing the same bottleneck, is explored.

- **Intra-Protocol fairness:** In this case, 6 TCP-ATL sources share the same bottleneck with a capacity of 6Mb/s for $d_w = 10$ ms and $p_w = 10^{-3}$. In Figure 55, the data received by each TCP-ATL sink is shown as a function of time. It is observed from Figure 55 that all sinks receive almost the same amount of data at any point in time during the connection period. Hence, the TCP-ATL sources are all given a fair share of the network resources.

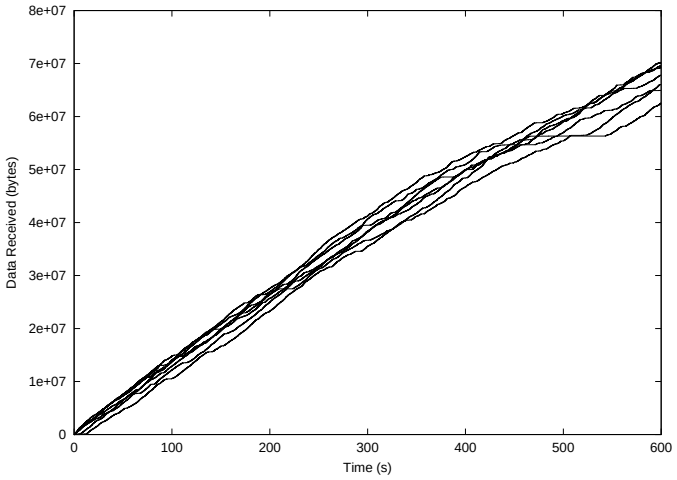

Figure 55: Intra-protocol fairness: The total bytes received by each of the TCP-ATL sinks with time for $d_w = 10$ ms and $p_w = 10^{-3}$.

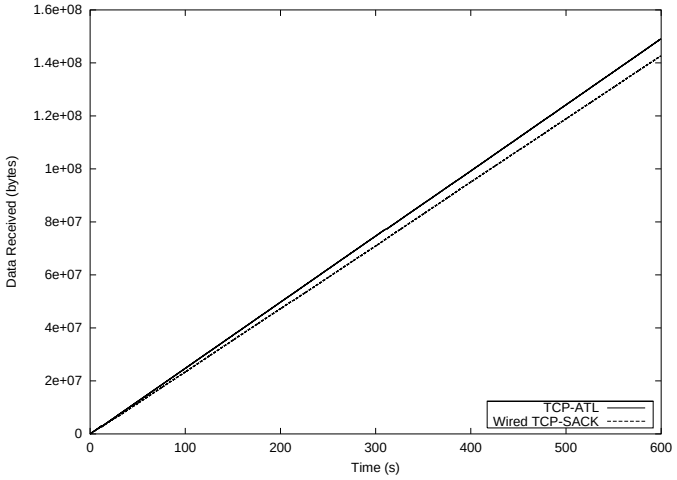

Figure 56: Fairness to wired TCP flows: The total bytes received by the TCP-ATL and wired TCP sinks with time for $d_w = 10$ ms and $p_w = 10^{-3}$.

- **Fairness to wired TCP flows:** In this case, 5 TCP-ATL and 5 wired TCP-SACK sources share the same bottleneck bottleneck with capacity 4Mb/s. TCP-ATL sources are connected to the bottleneck via a wireless link with $d_w = 10$ ms and $p_w = 10^{-3}$. TCP-SACK sources are connected to the bottleneck via a wired links, hence they are wired TCP source.

  The total data received by the TCP-ATL and wired TCP sinks are shown in Figure 56 as a function of time. Let $\Phi(t)$ be the ratio of the total data received by TCP-ATL and wired TCP sinks at time t. Here, although the TCP-ATL sink receives slightly higher

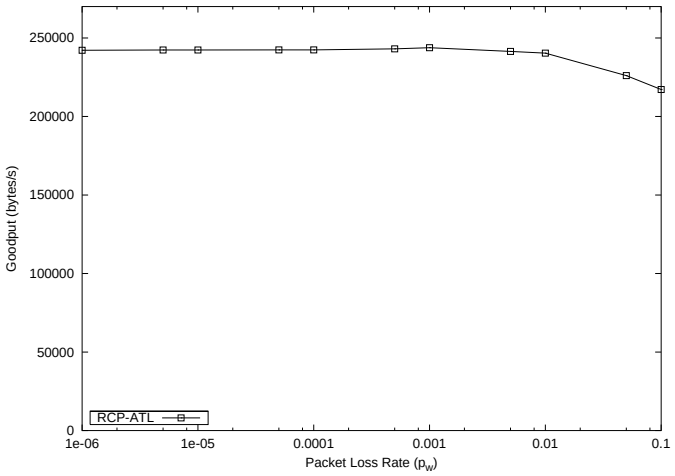

Figure 57: Goodput achieved by RCP-ATL for $d_w = 10$ ms and varying p_w.

amount of data than the wired TCP source, $\Phi(t) \approx 1$ throughout the connection period. This is mainly because of the inherent fairness of the TCP-ATL adaptive congestion control as explained in Section 4.3.2. Therefore, TCP-ATL significantly improves link utilization efficiency while preserving the overall fairness to the wired TCP sources sharing the same bottleneck.

4.5.2 RCP-ATL Performance

4.5.2.1 Goodput Performance

RCP-ATL goodput performance experiments are also performed for varying p_w between 10^{-6} and 10^{-1} and for $d_w = 10, 50, 150$ ms, which represent typical one-way wireless link delay values in WLAN, 3G and satellite networks, respectively. The simulation environment and parameters used are the same with TCP-ATL experiments presented in Section 4.5.1.1. In the literature, RAP [88] is the best rate control protocol for wired environments. However, it is shown in [106] that RAP has very poor performance in the wireless domain. In the literature, however, there is no rate control protocol at the transport layer specifically developed for WLAN and 3G cellular environments. On the other hand, RCS [106] is the rate control scheme that has the highest performance in satellite networks. Therefore, the RCP-ATL goodput performance is compared to only RCS for satellite networks.

For low p_w values, i.e., $p_w < 10^{-2}$ with $d_w = 10$ ms representing WLAN environments,

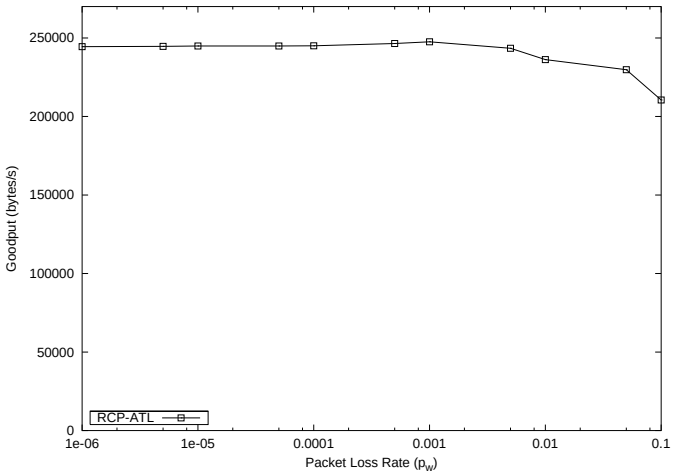

Figure 58: Goodput achieved by RCP-ATL for $d_w = 50$ ms and varying p_w.

RCP-ATL maintains its goodput performance as shown in Figure 57. As p_w increases, RCP-ATL starts to experience a slight performance degradation. As the wireless link delay increases, i.e., $d_w = 50$ ms, the goodput performance of RCP-ATL remains unaffected by the increasing delay as shown in Figure 58. This is achieved because of its adaptive rate control explained in Section 4.3.3.

In the typical satellite environment, i.e., $d_w = 150$ ms, the performance of RCP-ATL is also compared with RCS. In this case, RCP-ATL can maintain its high goodput performance over wide range of p_w while RCS experiences significant degradation for $p_w > 10^{-3}$ as shown in Figure 59. For $p_w = 10^{-2}$ and $d_w = 150$ ms, RCP-ATL improves goodput over RCS approximately by 228%. Therefore, RCP-ATL can achieve very high goodput performance in all different wireless environments as its TCP counterpart as shown in Figure 50, where α is now calculated by (49).

4.5.2.2 Fairness

Similar to the Section 4.5.1.4, two different scenarios, i.e., Intra-protocol fairness and Fairness to wired TCP flows, are considered for the fairness performance of RCP-ATL. In the intra-protocol fairness scenario, 6 RCP-ATL sources share the same bottleneck with capacity of 6Mb/s for $d_w = 10$ ms and $p_w = 10^{-3}$. As shown in Figure 60, the data received by each RCP-ATL sink is almost the same at any point in time during the connection period.

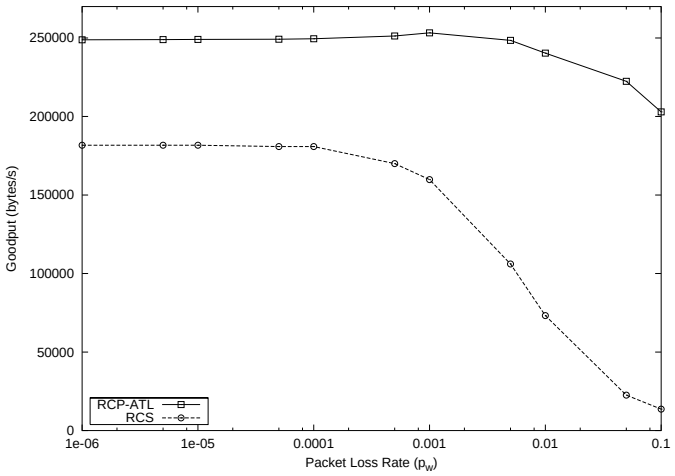

Figure 59: Goodput comparison of RCP-ATL, and RCS for $d_w = 150$ ms and varying p_w.

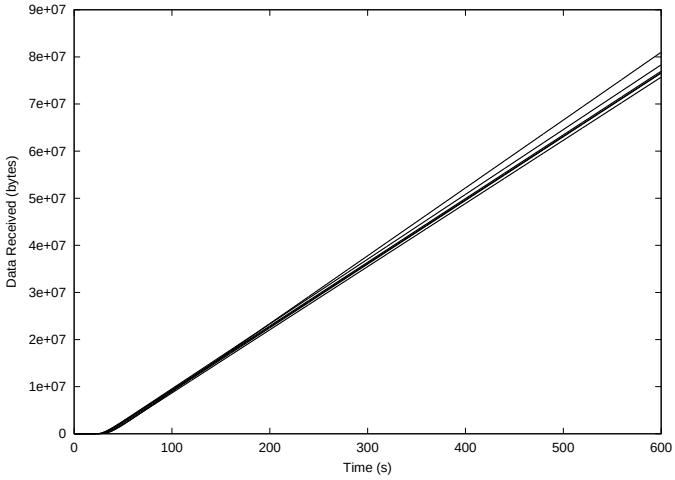

Figure 60: Intra-protocol fairness: The total bytes received by each of the RCP-ATL sinks with time for $d_w = 10$ ms and $p_w = 10^{-3}$.

Hence, the RCP-ATL sources are all given a fair share of the network resources.

To explore the RCP-ATL fairness to the wired TCP flows, the same scenario as in Section 4.5.1.4 is used. RCP-ATL and wired TCP-SACK sources share the same bottleneck with a capacity of 4Mb/s. The data received by the RCP-ATL and the wired TCP sinks are shown in Figure 61 as a function of time. Let $\Phi(t)$ be the ratio of the data received by RCP-ATL and wired TCP sinks at time t. As shown in Figure 61, $\Phi(t) \approx 1$ throughout the connection period. Therefore, RCP-ATL maintains fairness to the wired TCP sources sharing the same bottleneck. Note that this fairness is achieved because of the adaptive rate control designed by taking fairness into consideration as explained in Section 4.3.3.

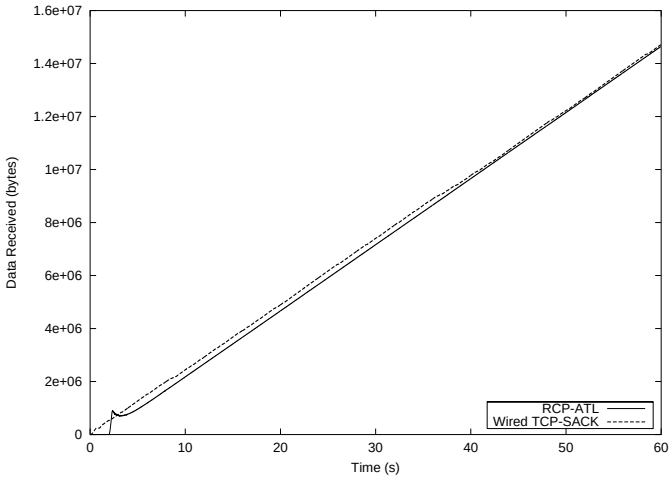

Figure 61: Fairness to wired TCP flows: The total bytes received by the RCP-ATL and wired TCP sinks with time for $d_w = 10$ ms and $p_w = 10^{-3}$.

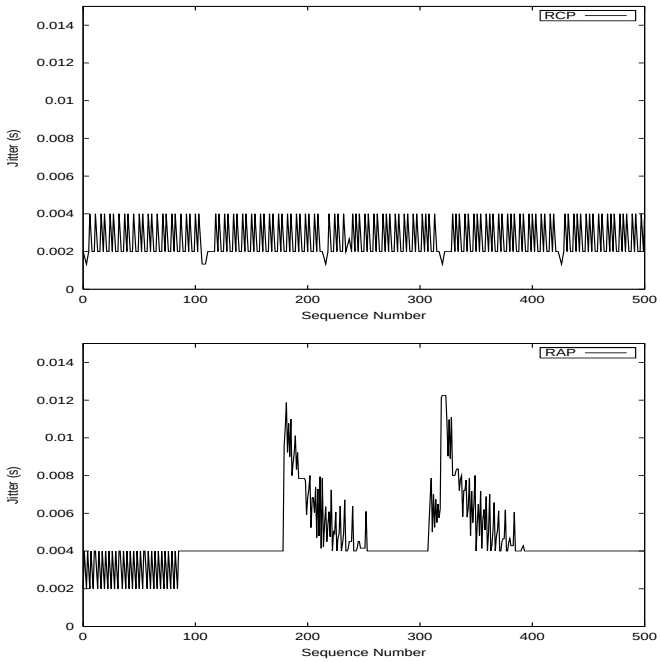

Figure 62: Jitter experienced by the RCP-ATL and RAP packets for $d_w = 10$ ms and $p_w = 10^{-4}$.

4.5.2.3 Jitter Issues

The jitter experienced by RCP-ATL and RAP flows are also investigated in terms of the multimedia support performance of the RCP-ATL. Although RAP [88] is mainly developed for wired networks, RCP-ATL is compared to RAP in order to assess the improvement

achieved by RCP-ATL due to its adaptivity to varying link conditions. The jitter experienced by each packet transmitted by the RCP-ATL and RAP sources are shown in Figure 62. In the bottom plot of Figure 62, it is seen that the packets sent by the RAP source experience much higher jitter than the packets sent by RCP-ATL. This is because RAP halves its data rate at packet losses due to link errors hence resulting in high delay variation. In order to comply with the instantaneous TCP-friendliness, RCP-ATL also performs rate decrease at each packet loss. However, since it uses higher multiplicative decrease parameter β, RCP-ATL experiences less abrupt rate variation in case of a packet loss than RAP. Therefore, RAP source experiences higher jitter than RAP source. Such improvement in jitter achieved by RCP-ATL, is of significant importance for the delay sensitive real-time traffic, where the delay variation directly affects the QoS performance.

CHAPTER V

RELIABLE TRANSPORT PROTOCOL FOR INTERPLANETARY INTERNET

In this chapter, a reliable transport protocol, TP-Planet, is presented for data traffic in the InterPlaNetary Internet. The objective of TP-Planet is to address the challenges and to achieve high throughput performance and reliable data transmission on deep space links of the InterPlaNetary Backbone Network. TP-Planet was first presented in [6]. In Section 5.2, an extensive set of related work as well as the performance of the existing TCP protocols are presented. TP-Planet protocol overview along with the detailed operation of the Initial State algorithm is presented in Section 5.3. In Section 5.4, Steady State algorithm including the new rate-based AIMD congestion control and Blackout State behavior are explained. Performance evaluation experiments and the results are then presented in Section 5.5.

5.1 Motivation

The space exploration missions are crucial for acquisition of information about the space and the universe. The entire success of a mission is directly related to the satisfaction of its communications needs. For this goal, the challenges posed by the InterPlaNetary Internet architecture described in Section 1.2 need to be addressed.

Among the architectural elements of the InterPlaNetary Internet shown in Figure 2, the main focus of this research is on the *Interplanetary Backbone Network where the source and sink end-points are basically the ground station at the Earth and the planetary gateways connected to the relay satellites orbiting around the outer-space planets.* This is because the Interplanetary Backbone Network plays a significant role for the performance of entire deep space communication. The most important characteristics and the challenges posed by the Interplanetary Backbone Network are listed as follows:

- **Very long propagation delays**: The deep space communication links may have

extremely long propagation delays. For example, the end-to-end round trip time for the Mars-Earth communication network varies from 8.5 minutes to 40 minutes according to the orbital location of the planets [44].

- **High link error rates**: The bit error rates on the deep space links are very high usually on the order of 10^{-1} [44].
- **Blackouts**: Periodic link outages may occur due to orbital obscuration with the loss of line-of-sight because of moving planetary bodies, the interference of an asteroid or a spacecraft [22].
- **Bandwidth asymmetry**: The asymmetry in the bandwidth capacity of forward and reverse channels is typically on the order of $1000 : 1$ in space missions [44].

These challenges need to be addressed in order to meet the communication requirements of deep space missions and hence realize the InterPlaNetary Internet objective.

5.2 Related Work

The existing TCP variants [65], [66], [51], [81], [28], [80], [31], [11] have been shown to achieve very poor performance in deep space communication networks [2]. The dominant factor in this performance degradation is the extremely high propagation delay in deep space links [2]. This is solely due to the window-based mechanism used by the current TCP protocols during slow start and congestion avoidance algorithms. In the slow start algorithm, the congestion window size (W) is increased by one packet per received ACK until the slow start threshold (W_{ss}) is reached, i.e., $W < W_{ss}$. However, this approach wastes the link resources for a very long duration which is proportional to the propagation delay. For $W_{ss} = 20$ and $RTT = 20$ minutes, it is shown in [2] that the slow start algorithm cannot utilize the link resources for approximately 120 minutes in deep space links.

The inefficiency in link utilization due to window-based mechanisms also exists during the congestion avoidance phase, i.e., $W \geq W_{ss}$, where the TCP source increments the congestion window size by roughly one at each RTT. As shown in Figure 63, the existing window-based TCP protocols achieve throughput of approximately 10 bytes/s for the link

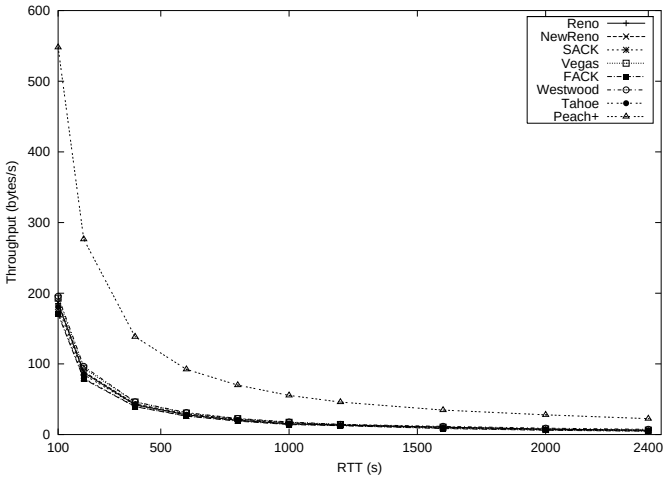

Figure 63: Throughput performance of TCP implementations over InterPlaNetary Backbone links.

capacity of 1 Mb/s, packet loss probability of $p = 10^{-3}$ and $RTT = 40$ minutes. In other words, the entire deep space link remains almost unutilized during the entire connection period. Note that $RTT = 40$ minutes is within the RTT range for communication links between Mars and Earth, i.e., 8.5 to 40 minutes based on the orbital position [44].

Furthermore, the current TCP protocols are designed for wired links, which are reasonably assumed to have negligible bit error rates. Therefore, they invoke congestion control mechanisms in case of a single packet loss. However, this assumption does not hold in deep space communication links. Consequently, the packet loss based congestion detection mechanism results in unnecessary rate throttle and leads to severe throughput degradation. Much research has been performed in recent years in order to address the throughput degradation due to wireless link errors [19]. However, these solutions cannot be directly applied to Interplanetary Backbone Network because of the amplifying effects of the extremely high propagation delay and the other abovementioned characteristics on the problem.

The TCP performance on the links with high bandwidth-delay products and errors is analyzed in [72]. Many transport protocols [57], [10], [11] are proposed for satellite links, which are also characterized by high bandwidth-delay products and high bit error rates. Nevertheless, these studies mostly refer to Geo-stationary Earth Orbit (GEO) satellite links with typical RTT values around 550 ms, which are very low compared to RTTs in deep space

communication links. Moreover, packet losses due the blackout conditions may also mislead the congestion control mechanisms based on packet losses. In [55], an enhancement for TCP is developed to address signal loss conditions due to mobility. However, the blackout situations in deep space links are much more complicated due to extremely high propagation delay and hence solutions as in [55] cannot be applied directly.

There already exists an active research on transport layer protocols for space-based communication networks. Space Communications Protocol Standards-Transport Protocol (SCPS-TP) [43], [38] is a set of TCP extensions developed by Consultative Committee for Space Data Systems (CCSDS) for space communications. SCPS-TP is designed to support current communication environments and those of upcoming space missions [38]. SCPS-TP is developed based on the existing TCP protocols with some modifications and extensions to address the challenges posed by space-based systems such as link errors, bandwidth asymmetry, and link outages. It can provide full, best-effort and minimal reliability according to the mission specific communication requirements. The capabilities of the SCPS-TP are basically a combination of existing TCP protocols, which are shown to be inadequate in addressing the challenges in Interplanetary Backbone Network [2]. For example, SCPS-TP with Vegas congestion control uses window-based scheme and adopts slow-start algorithm. Although the rate-based version of SCPS-TP is under development, it disables congestion control mechanism and performs transmission with user selected fixed rate [113]. On the other hand, for example, SCPS-TP uses TCP-Vegas [28] congestion decision mechanism based on the RTT variation. However, since the window-based nature of TCP-Vegas cannot fully utilize the link, it is not even possible for it to experience the congestion and hence variation in RTT. Therefore, congestion decision based on RTT variation does not provide proper congestion control functionality. Furthermore, due to the extremely high propagation delay, the variation in RTT may not be measured accurately such that the resultant congestion control behavior may also not be accurate.

As pointed out in [2], there exists an urgent need for reliable transport protocol for InterPlaNetary Internet. In this chapter, a reliable transport protocol, TP-Planet, for

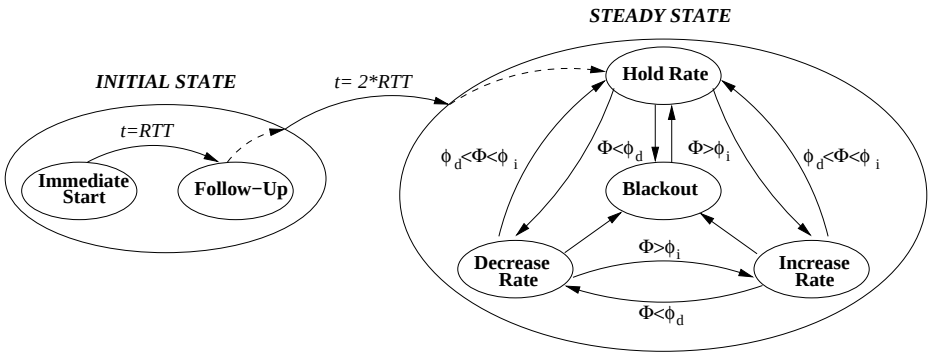

Figure 64: TP-Planet protocol operation state diagram including substates and the state transitions based on congestion control decision mechanism.

InterPlaNetary Internet (IPN) is presented. TP-Planet is mainly implemented at the Interplanetary Backbone Network nodes, i.e., the TP-Planet source and sink are the ground station at the Earth and the planetary gateway connected to the relay satellites orbiting around the outer-space planets. The objective of TP-Planet is to achieve high throughput performance and reliable data transmission by addressing the challenges in the InterPlaNetary Backbone Network. In order to address the challenges due to extremely high propagation delay, TP-Planet deploys a newly developed end-to-end rate-based additive-increase multiplicative-decrease (AIMD) congestion control, whose AIMD parameters are adjusted to compensate for the throughput degradation. Two novel algorithms, i.e., Initial State and Steady State, constitute the structure of the TP-Planet protocol. Initial State algorithm replaces the inefficient slow start algorithms in order to capture link resources in a very fast and controlled manner. In Steady State, a new congestion detection and control mechanism is deployed to minimize the erroneous congestion decisions due to high link errors. In order to reduce the effects of blackout conditions on the throughput performance, TP-Planet incorporates Blackout State procedure into the protocol operation. Bandwidth asymmetry problem is addressed by the adoption of delayed SACK options. Performance evaluation via simulation experiments reveals that TP-Planet significantly achieves high throughput performance and addresses the challenges in InterPlaNetary Backbone Network.

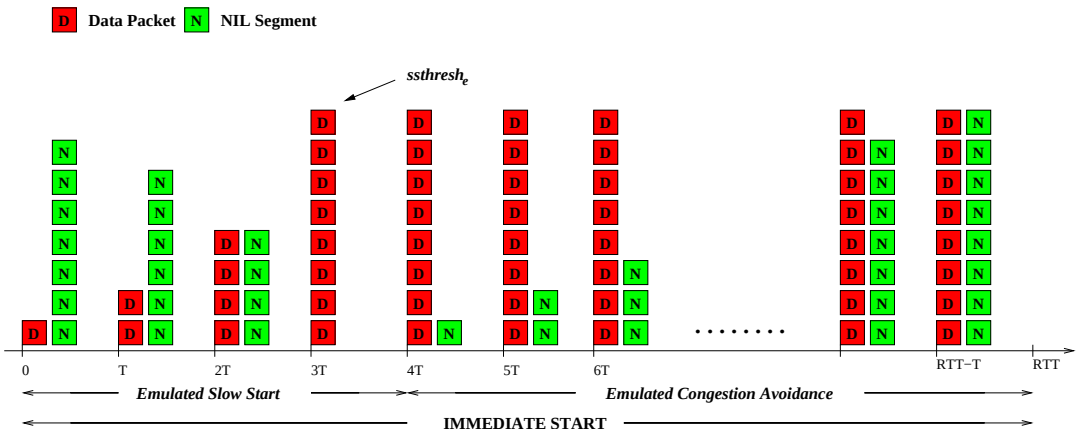

Figure 65: An example illustration of time framing mechanism in the Immediate Start phase for $ssthresh_e = 8$.

5.3 TP-Planet: Initial State

TP-Planet source starts connection in the *Initial State* at $t = 0$ by calling `Initial_State()` algorithm as shown in Figure 64. TP-Planet source then goes to the *Steady State* by calling `Steady_State()` algorithm at $t = 2 \cdot RTT$ as shown in Figure 64.

The slow start algorithm used in the existing TCP protocols is shown to be inefficient on deep space links of the InterPlaNetary Internet [2]. In order to avoid performance degradation due to Slow Start algorithm, TP-Planet deploys `Initial_State()` algorithm, which achieves to capture link resources in a very fast and controlled manner. The algorithm is composed of two main procedures, i.e., *Immediate Start* and *Follow-Up*.

5.3.1 Immediate Start

TP-Planet starts a connection in the *Immediate Start* state at $t = 0$, where the actual RTT is divided into time intervals of size T. TP-Planet then emulates Slow Start and Congestion Avoidance algorithms of the conventional window-based TCP protocols by treating time intervals of size T as the RTT of the emulated connection. The objective of the Immediate Start phase is to probe the network in a fast and controlled manner so that the transmission rate can be increased quickly according to the feedback from the sink. The determination of the interval size T is presented in Section 5.3.3. Here, the protocol operation in the Immediate Start phase is presented in two parts, i.e., *Emulated Slow Start* and *Emulated*

Congestion Avoidance.

5.3.1.1 Emulated Slow Start

As shown in Figure 65, the first RTT is divided into time intervals of size T. This is done in order to have more fine-grained time unit than the extremely high RTT of the deep space link. The number of data packets transmitted in each interval, *cwnd*, is increased geometrically in each interval by emulating window-based behavior of the Slow Start algorithm of the classical TCP protocols. This increase continues until the emulated slow start threshold, $ssthresh_e$, is reached, i.e., $cwnd \leq ssthresh_e$.

In addition to data packets, *NIL segments* [11] are also sent in each time interval during the Immediate Start phase. The objective of NIL segment transmission is to probe actual link resource status in the beginning of the connection. NIL segments are chosen from the unacknowledged outstanding data packets to be used for packet loss recovery at the receiver. NIL segments are encapsulated by low priority IP packets. Note that the *type of service* (TOS) field of the IP packet header can be used for this purpose. Hence, assuming the relay satellites serving as routers along the Interplanetary Backbone link as shown in Figure 2 have priority-queuing capability, low priority NIL segments are discarded first in case of congestion. Therefore NIL segment transmission does not affect the throughput of the actual data packet transmission. If there is no congestion, they are received and ACKed back by the sink, which reveals that there are still unutilized link resources along the path. NIL segments are created with `Generate_Nil_Segment()` algorithm shown in Figure 66, where Q is the length of the queue of unacknowledged outstanding data packets and i is a counter. Further details of the NIL segment generation can be found in [11].

In the Emulated Slow Start phase, the number ($cwnd_N$) of NIL segments transmitted in each interval is determined such that the total number of packets transmitted does not exceed the emulated slow start threshold, i.e., $cwnd + cwnd_N \leq ssthresh_e$.

In Figure 65, the emulated slow start threshold is assumed to be 8 packets, i.e., $ssthresh_e = 8$. Therefore, the number of data and NIL segments transmitted at each time interval during the Emulated Slow Start phase can be summarized as follows

```
Generate_NIL_Segment()
  Add unACKed packets into the queue;
  i = 0;
  if (NIL_Segment is needed)
    q = i mod Q;
    i = i + 1;
    NIL_Segment = q^th packet in queue;
  end;
  if (j^th packet in the queue is ACKed)
    Remove the packet from Q;
    if (j < i and i > 0)
      i = i - 1;
    end;
  end;
  if (New packet is added to Q)
    Add the packet to the tail;
    Q = Q + 1;
  end;
  Return NIL_Segment;
end;
```

Figure 66: The NIL Segment Generating Algorithm

- 1^{st} T $cwnd = 1$ and $cwnd_N = 7$.
- 2^{nd} T $cwnd = 2$ and $cwnd_N = 6$.
- 3^{rd} T $cwnd = 4$ and $cwnd_N = 4$.
- 4^{th} T $cwnd = 8$ and $cwnd_N = 0$.

In general, the number of data packets and NIL segments transmitted in the i^{th} time interval during the Emulated Slow Start is given by

$$cwnd = 2^{i-1} \tag{54}$$

$$cwnd_N = ssthresh_e - 2^{i-1} \tag{55}$$

5.3.1.2 Emulated Congestion Avoidance

The Emulated Slow Start is left for Emulated Congestion Avoidance phase when $cwnd = ssthresh_e$. Since there is no feedback information about the link resources as yet, the TP-Planet source does not increase the number $cwnd$ of data packets transmitted in each time interval. Therefore, $cwnd = ssthresh_e$ until the end of the emulated congestion avoidance

phase. However, for further probing the link status, the source increases $cwnd_N$ additively by one NIL segment per T interval emulating congestion avoidance algorithm of the classical TCP protocols. As in Figure 65, during this phase *cwnd* is kept constant and $cwnd_N$ is increased by one segment in each T until the end of the Immediate Start, when $cwnd_N$ becomes equal to *cwnd*. The transmission of NIL segments is terminated at the end of the first RTT and TP-Planet source goes to the *Follow-Up* state of the Initial State algorithm as shown in Figure 64.

5.3.2 The Follow-Up Phase

During this phase, the feedback for the packets transmitted in the Immediate Start phase are started to be received by the TP-Planet source. In order to save scarce reverse channel resources of the bandwidth asymmetrical deep space link, the sink does not ACK back all the packets it receives . Several data packets are ACKed by a delayed SACK, whose details are explained in Section 5.4.4. Since NIL segments are transmitted to probe the link status, each received NIL segment indicates the existence of unutilized link resources. Hence, the TP-Planet sink counts the total number (N) of packets received in every T period and sends this information back to the source. This information is carried in NIL ACKs. TP-Planet source transmits $ssthresh_e$ packets in $RTT \leq t \leq RTT+T$ until the first NIL ACK received at $t = RTT + T$. Then, TP-Planet source adjusts its transmission rate (S) by using the information carried in NIL ACKs, i.e., $S = N/T$, until the end of $2 \cdot RTT$. Therefore, the transmission rate S at the end of the Initial State depends on the number of ACKs received in the last time interval of the Follow-Up phase.

Let N_{ACK} be the number of packets received by the TP-Planet sink during $RTT - T \leq t \leq RTT$. Since the total number of packets sent in the last interval of the Immediate Start phase is $2 \cdot sstresh_e$, the data transmission rate S at the end of the Initial State is expressed by

$$S = \frac{\min\{N_{ACK}, 2 \cdot ssthresh_e\}}{T} \tag{56}$$

In addition to the data packets, TP-Planet source also starts sending *NIX segments* during the Follow-Up phase. The NIX segments are much smaller than the data packets,

i.e., 40 bytes. They are carried in both low and high priority IP packets. During the Follow-Up phase, low and high priority NIX segments are transmitted with the transmission rate S_{Nix} which is equal to the transmission rate of the data packets, i.e., $S_{Nix} = S$. The objective of the NIX segment transmission is to capture congestions and make decisions accordingly in the Steady State. The details of the congestion detection mechanism based on NIX segments are explained in Section 2.5.2. The Follow-Up phase is over at $t = 2 \cdot RTT$ and the source leaves Initial State for the Steady State as shown in Figure 64.

```
Initial_State()
  Send connection request CONN_REQUEST;
  Set T & ssthresh_e by (61) & (62)
  n = 1;
  cwnd = 1;
  While (t ≤ RTT)
  /* Immediate Start */
    While (cwnd ≤ ssthresh_e)
    /* Emulated Slow Start */
      If ((n − 1)T ≤ t ≤ nT)
        Send (cwnd) DATA pkts;
        Send (ssthresh_e − cwnd) NIL pkts;
      end;
      n = n + 1;
      cwnd = 2^(n−1);
    end;
    /* Emulated Congestion Avoidance*/
    cwnd = ssthresh_e;
    cwnd_N = 1;
    While ((n − 1)T ≤ t ≤ nT ≤ RTT)
      Send (cwnd) DATA pkts;
      Send (cwnd_N) NIL pkts;
      cwnd_N = cwnd_N + 1;
      n = n + 1;
    end;
  end;
  While (RTT ≤ t ≤ 2 · RTT)
  /* Follow-Up */
    If (NIL_ACK_RECEIVED)
      Set data rate S = N/T;
      Set NIX rate S_Nix = S;
    end;
  end;
  Steady_State();
end;
```

Figure 67: The `Initial_State()` algorithm

Consequently, TP-Planet source can capture the link resources as soon as $t \geq RTT + T$

based on the number of ACKs it receives from the sink. It can achieve this without leading to congestion by controlling the number of probe packets injected into the network.

5.3.3 Determination of the Time Interval T

In the Initial State, the extremely high actual RTT is divided into time intervals of size T and the conventional TCP behavior is emulated to capture the available link resources in a controlled manner. The performance of the Initial State depends on the size of the T intervals. While the smaller T increases link utilization, it also increases overhead incurred by NIL segment transmission.

The objective of the Initial State is to reach a certain data transmission rate, S, as soon as possible without leading to a congestion. During the Follow-Up phase, TP-Planet source adjusts its transmission rate S according to the feedback received from the sink every T period via NIL ACKs. Since $ssthresh_e$ is the maximum number of packets transmitted in one interval during the Emulated Slow Start phase, the transmission rate reached in the Follow-Up phase is dependent on $ssthresh_e$. Here, it is assumed that the TP-Planet source has a given target minimum transmission rate requirement, B. This requirement can be mostly due to two main reasons:

- **Application**: The minimum transmission rate is required in order to meet specific application needs such as transmission of scientific multimedia data which requires 100% reliability and a minimum bound on the transmission rate.
- **Latency**: The maximum allowed latency can be advertised by the specific space mission requirements as the maximum duration for the transmission of certain amount of data. This determines the minimum target transmission rate requirement for a given connection.

Consequently, in order to achieve the target transmission rate of B packets/s at $t = RTT + T$, the corresponding $sshtresh_e$ is calculated as

$$ssthresh_e \quad = \quad B \cdot T \tag{57}$$

The Emulated Slow Start phase of the Immediate Start terminates when the number of data packets transmitted in one time interval reaches the emulated slow start threshold, i.e., $cwnd = ssthresh_e$. Therefore, from (55) it follows that the duration of the Emulated Slow Start phase (T_{ess}) is given by

$$T_{ess} \quad = \quad (\log_2 ssthresh_e + 1) \cdot T \tag{58}$$

During the Emulated Congestion Avoidance phase, the number of data packets per interval, $cwnd$, is kept constant. Moreover, the number of NIL segments is increased by one per interval until the end of the Immediate Start, i.e., $t = RTT$. In order to probe the network resources in the range of $[B, 2B]$, the source increases the number of NIL segments transmitted in each interval until it is equal to the number of data packets, i.e., $cwnd_N = ssthresh_e$. The Immediate Start is over when $cwnd_N$ reaches $ssthresh_e$ at $t = RTT$. Therefore, the duration of the Emulated Congestion Avoidance phase (T_{eca}) can be expressed by

$$T_{eca} \quad = \quad ssthresh_e \cdot T \tag{59}$$

Since the total duration of the Immediate Start is $T_{ess} + T_{eca} = RTT$, from (58) and (59) it follows that

$$(\log_2 ssthresh_e + 1 + ssthresh_e) \cdot T \quad = \quad RTT \tag{60}$$

As $(\log_2 ssthresh_e + 1)$ is negligible compared to $ssthresh_e$ itself, from (57) and (60) the size of the time interval T can be calculated by

$$T \quad = \quad \sqrt{\frac{RTT}{B}}\,, \tag{61}$$

where B is the target transmission rate in packets/s. At the beginning of a connection, RTT is also not known to the TP-Planet source. Therefore, TP-Planet source also caches the RTT of the past connections and uses that value in order to calculate the time interval size T by (61) during the Initial State.

Consequently, the emulated slow start threshold can be expressed by using (57) and (61) as follows

$$ssthresh_e \quad = \quad \sqrt{RTT \cdot B} \tag{62}$$

Thus, at the beginning of the connection, T and $ssthresh_e$ are set by (61) and (62). The first RTT period is then divided into time intervals of size T. The Emulated Slow Start and Emulated Congestion Avoidance phases are performed in the first RTT in order to perform resource probing. In the second RTT, the transmission rate is increased by sending a new data packet for each received ACK until $t = 2 \cdot RTT$. By this way, TP-Planet source can increase its transmission rate very quickly and efficiently utilize the resources during the Initial State without leading to any congestion.

5.3.4 The Connection Establishment

The conventional TCP protocols perform three-way handshake for connection establishment. The existing protocols send connection request segment at the beginning of the connection and do not transmit any data segments until the connection ACK is received from the receiver. This approach results in waste of huge bandwidth for at least a duration of one RTT in deep space links. In order to avoid this inefficiency, TP-Planet source does not wait for the ACK for the connection request and starts data transmission in the Initial State as if the session request is granted. If the request is rejected, then the connection is terminated after one RTT.

As a result, TP-Planet achieves better utilization of link resources in the early phases of the connection by the Initial State algorithm. The overall Initial State algorithm of the TP-Planet is summarized in Figure 67.

5.4 TP-Planet: Steady State

TP-Planet source leaves Initial State for *Steady State* at $t = 2 \cdot RTT$ and remains in the Steady State until the connection is terminated. During the Steady State operation, TP-Planet source can be in one of the four states, i.e., *Increase Rate*, *Decrease Rate*, *Hold Rate* and *Blackout* as shown in Figure 64. In the beginning of the Steady State, the source goes to Hold Rate state, where no transmission rate change is performed. During Steady State operation, TP-Planet deploys a new congestion control scheme. Hence, the transitions between these states in the Steady State is decided based on this congestion control scheme. Therefore, the data transmission rate S can be *increased*, *decreased* or *hold* according to the

current state.

5.4.1 Congestion Control

TP-Planet deploys a new congestion control scheme in order to address the challenges due to high link error rates in Interplanetary Backbone Network. The objective of this method is to decouple network congestions and packet losses due to errors.

During the Steady State, TP-Planet source transmits low and high priority *NIX segments* continuously for congestion detection purposes. NIX segments are carried by IP packets, which are marked as *high* and *low* priority using TOS field in the IP packet header. NIX segments differ from NIL segments used in Initial State in terms of their size and functionality. Unlike NIL, NIX segments are much smaller compared to data segments, i.e., 40 bytes. They do not carry any information and thus, cannot be used for error recovery purposes.

Low and high priority NIX segments are transmitted simultaneously with the same rate (S_{Nix}) equal to the data transmission rate (S), i.e., $S_{Nix} = S$. The objective of this is to obtain congestion decision support via comparison between the reception statistics of both low and high priority NIX segments. Since low and high priority NIX segments are equal in size and are transmitted with the same rate S_{Nix}, they experience the same packet loss rate due to space link errors. However, assuming the relay satellites serving as routers along the Interplanetary Backbone link as shown in Figure 2 have priority-queuing capability, low priority NIX segments are discarded first in case of a congestion. The only reason for low and high priority NIX segments to have different packet loss rates is the additional loss experienced by low priority segments due to congestion. This reasoning constitutes the basis for the congestion detection method via NIX segment transmission.

TP-Planet sink counts the number of received low (N_{Low}) and high (N_{High}) priority NIX segments in a sliding time window of T_w. The received NIX segments are not ACKed back to the sender. Note that this also avoids an overhead in the reverse channel, which can also become a bottleneck due to bandwidth asymmetry in deep space links. Only the reception statistics within a time window of T_w are returned to the sender every τ period.

This information is carried by *NIX ACKs*. TP-Planet source receives NIX ACKs carrying (N_{Low}, N_{High}) at the end of each measurement period τ.

Let Φ be the ratio of the number of received low and high priority NIX segments, i.e.,

$$\Phi = \frac{N_{Low}}{N_{High}} \tag{63}$$

Assuming that the low and high priority NIX packets experience same packet loss rate due to the link errors within a measurement period τ, since the only reason for $N_{Low} < N_{High}$ is the congestion along the path, TP-Planet source infers that a congestion exists if $\Phi < 1$.

The entire congestion control of the TP-Planet source is determined according to the value of Φ. The transitions between *Increase*, *Decrease* and *Hold* Rate states in Figure 64 are performed based on Φ. In order to avoid unnecessary state transitions, decision is made via comparison of Φ with preset rate decrease, ϕ_d, and the rate increase thresholds, ϕ_i, as shown in Figure 64. These thresholds are protocol parameters, whose selection is an implementation issue. The values of ϕ_d and ϕ_i, which yield highest protocol performance, are determined during simulation experiments and are given in Section 5.5.

The summary of the congestion control mechanism is given as follows:

1. $\mathbf{\Phi < \phi_d}$: In this case, TP-Planet source infers that congestion is experienced along the path. Thus, the source goes to the *Decrease Rate* state where the transmission rate S is decreased multiplicatively, i.e., $S = S \cdot \zeta$.

2. $\mathbf{\phi_d \leq \Phi \leq \phi_i}$: In this case, the data transmission rate S is kept unchanged until next feedback is received from the sink.

3. $\mathbf{\Phi > \phi_i}$: TP-Planet source infers that no congestion is experienced. Consequently, it increases the data transmission rate additively, i.e., $S = S + \delta$.

The additive-increase (δ) and multiplicative-decrease (ζ) parameters are used to perform AIMD rate control every τ period. The selection of these AIMD parameters are explained in Section 5.4.2. The steps of the Steady State algorithm of TP-Planet protocol is summarized in Figure 68.

```
Steady_State()
  Set ξ;
  Set α by (65);
  Set ζ & δ by (66) & (67);
  Send DATA pkts with rate S;
  Send Low pri.  NIX pkts with rate S;
  Send High pri.  NIX pkts with rate S;
  If (NIX_ACK_RECEIVED)
  Congestion Decision:
    If (Φ < φ_d)
    /* Decrease Rate */
      S = S · ζ;
    else if (φ_d ≤ Φ ≤ φ_i)
    /* Hold Rate */
      S = S;
    else if (Φ > φ_i)
    /* Increase Rate */
      S = S + δ;
    end;
  end;
  If (ZERO_NIX_ACK_RECEIVED)
    /* After Blackout State */
        Goto Hold State;
  end;
  If (NO_ACK_in_T_w)
    /* Blackout State */
    Send Low pri.  NIX pkts with rate S;
    Send High pri.  NIX pkts with rate S;
    Retransmit TIMEOUT pkts;
    If (NIX_ACK_RECEIVED)
      If (ZERO_NIX_ACK)
        /* L ≥ 2x */
        Goto Hold State;
      else
        /* L < 2x */
        Goto Congestion Decision;
      end;
    If (DATA_ACK_RECEIVED)
      /* L < 2x */
      Goto Congestion Decision;
    end;
  end;
end;
```

Figure 68: The `Steady_State()` algorithm.

5.4.2 The New Rate-Based AIMD Scheme

As mentioned before, current TCP protocols achieve very poor performance on the links with extremely high propagation delay mostly due to their window-based operation [2].

The throughput of the window-based TCP protocols and rate-based schemes are inversely proportional to the RTT [83] and the square-root of RTT [8], respectively. Thus, the rate-based congestion control schemes are more robust to excessive propagation delays than the window-based mechanisms. Hence, in order to address the adverse effects of extremely high propagation delay on the throughput performance, TP-Planet deploys rate-based additive-increase multiplicative-decrease (AIMD) congestion control. The steady state throughput of the rate-based AIMD scheme can be expressed by [8]

$$T = \frac{\alpha}{4 \cdot (1-\xi)} \left[1 + \xi + \sqrt{(3-\xi)^2 + \frac{8 \cdot (1-\xi^2)}{\alpha \cdot RTT \cdot p}} \right] \tag{64}$$

where α and ξ are the additive-increase and the multiplicative-decrease factors, respectively and p is the packet loss probability. It is observed from (64) that the throughput of the rate-based AIMD scheme depends on the values of α and ξ. Therefore, TP-Planet adapts its AIMD parameters to the link conditions, i.e., RTT and packet loss rate, such that their adverse effect on the performance are compensated and a given target throughput is achieved.

The additive-increase parameter (α) of the rate-based AIMD scheme to achieve a target throughput of B packets/s can be obtained from (64) as follows

$$\alpha = \frac{(1+\xi)}{2} \left(B + \frac{1}{RTT \cdot p} \right) \left[\sqrt{1 + \frac{8B^2(1-\xi)}{\left(B + \frac{1}{RTT \cdot p} \right)^2 (1+\xi)^2}} - 1 \right] \tag{65}$$

where ξ is the multiplicative-decrease factor and p is the packet loss probability, respectively. Here, the target throughput B can be defined as the average data rate required to transmit certain amount of information within the certain delay bound as in Section 5.3.3.

These AIMD parameters, α and ξ, in (65) are the rate change parameters to be used to control the rate with period RTT. However, TP-Planet source performs rate control with the feedback received from the sink every τ period, where $\tau \ll RTT$. Thus, the rate control parameters α and ξ in (65) cannot be directly used for TP-Planet. These parameters, instead, represent the upper bound for the rate change of the TP-Planet source within one RTT period. Hence, the AIMD parameters to be used by the source with period τ can be derived from α and ξ.

Let τ be the NIX segment reception statistics feedback period. Thus, TP-Planet source performs at most RTT/τ number of rate changes within one RTT period.

Let δ and ζ be additive-increase and multiplicative-decrease parameters to be used by TP-Planet for rate control with period τ. Then ζ can be calculated as

$$\zeta = \xi^{(\tau/RTT)} \tag{66}$$

By the same reasoning, the additive increase factor to be used by TP-Planet source can also be calculated as

$$\delta = \frac{\alpha \cdot \tau}{RTT} \tag{67}$$

TP-Planet source uses AIMD scheme during the Steady State operation as shown in Figure 64. At the end of the Initial State, i.e., $t = 2 \cdot RTT$, it calculates the packet loss rate p by using the number of transmitted and ACKed data packets and NIL segments. TP-Planet source then uses (65), (66) and (67) to calculate its AIMD parameters, i.e., ζ, δ. Consequently, according to the result of the congestion decision mechanism presented in Section 5.4.1, the transmission rate S is multiplicatively decreased or additively increased with ζ and δ, respectively.

5.4.3 The Blackout State Behavior

Link outages due to loss of line-of-sight by orbital obscurations lead to burst packet losses and decrease in the throughput. In order to provide reliable transport, SACK options [81] are adopted by TP-Planet to address burst losses. Due to possible inadequacy of the number of SACK blocks in the SACK option field for very long blackout durations, timeout mechanism is also included in TP-Planet. In order to reduce the throughput loss due to blackouts, *Blackout State* is developed and incorporated into the Steady State.

TP-Planet source receives data ACKs for reliability control purposes and NIX ACKs for NIX segment reception statistics. If the source does not receive any type of ACK for a certain period of time T_w, it infers this condition as blackout and goes to the *Blackout State* as shown in Figure 64. The objective of the Blackout State procedure is to reduce the throughput degradation due to the blackout situation.

During blackout, TP-Planet source keeps sending low and high priority NIX packets without changing its transmission rate. The same blackout event is also detected by the TP-Planet sink if no packet is received within T_w period. Although the sink does not receive any low and high priority NIX packets during the blackout, it keeps sending NIX ACKs with (N_{Low}, N_{High}) as (0,0). These ACK packets are called *Zero NIX ACKs*. The objective of Zero NIX ACKs is to help TP-Planet source to capture accurate information regarding the blackout situation and act accordingly.

Since RTT is very high, the effect of blackout on the performance changes with its relative location of blackout occurrence with respect to the sink. Let $t = t_0$ be the time when blackout occurs and L is the duration of the blackout. Assume that the blackout occurs at a position x seconds away from the TP-Planet sink. For $rtt = RTT/2$, there are two distinct cases according to the duration of the blackout and its relative distance to the TP-Planet sink in time:

1. $L < 2x$: After $rtt - x$ from t_0, i.e., at $t_1 = t_0 + rtt - x$, TP-Planet source detects the period without ACKs. If the duration of the period with no ACK takes more than T_w, then the source moves to Blackout State at $t = t_1$, as in Figure 69.

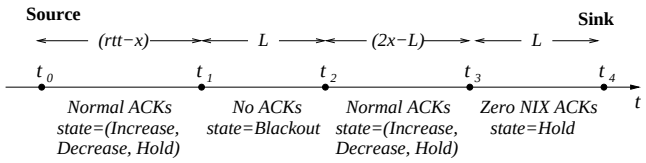

Figure 69: Blackout condition observed from TP-Planet source for $L < 2x$.

In this case, TP-Planet source does not send any new data packets but sends low and high priority NIX segments without changing their transmission rate. It also does not invoke congestion control mechanism. It starts retransmission of the packets whose retransmission timer is expired. At $t_2 = t_1 + L$, TP-Planet source receives normal ACKs for duration of $2x - L$. Therefore, TP-Planet source infers that blackout is over and goes to any of the Increase, Decrease and Hold states according to the information it receives in the first NIX ACK as shown in Figure 64. At $t_3 = t_2 + 2x - L$, the source starts to receive Zero NIX ACKs for duration of L. These Zero NIX ACKs,

are in fact, transmitted by the receiver when it detects the same blackout condition. Therefore, TP-Planet does not go to Blackout State instead it goes to Hold State, where it keeps sending new data packets with the same transmission rate again as shown in Figure 64. Consequently, TP-Planet reduces the effect of blackout on the performance by not wasting the link resources for a duration of L.

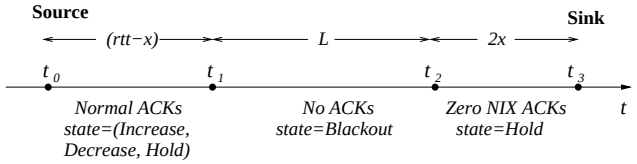

Figure 70: Blackout condition observed from TP-Planet source for $L \geq 2x$.

2. $L \geq 2x$: In this case, TP-Planet source detects no ACK period and goes to Blackout State at $t_1 = t_0 + rtt - x$ for the blackout which occurred at $t = t_0$. It again does not change its transmission rate and does not send any new data packet. At $t_2 = t_1 + L$, the source starts to receive zero NIX ACKs for duration $2x$ as shown in Figure 70. This reveals that the blackout is over for TP-Planet source and then the source leaves Blackout State for Hold State as shown in Figure 64. It stays in Hold State and keeps sending data packets with the same transmission rate before the blackout was detected. At $t_3 = t_2 + 2x$, Zero NIX ACKs period is over and according to the information received in the first NIX ACK the source performs state transition. Hence, the duration of $2x$ is efficiently utilized by the help of Zero NIX ACK mechanism.

Consequently, the Blackout State reduces the throughput degradation due to blackout conditions and improves the link utilization for duration of L or $2x$ in the cases $L < 2x$ and $L \geq 2x$, respectively.

5.4.4 The Delayed SACK

TP-Planet uses selective acknowledgement (SACK) [81] options for assurance of reliable data segment transmission. TP-Planet sink continuously sends SACK back to the source for each data packet it receives. Given that the data packets are 1KB and SACKs are 40B,

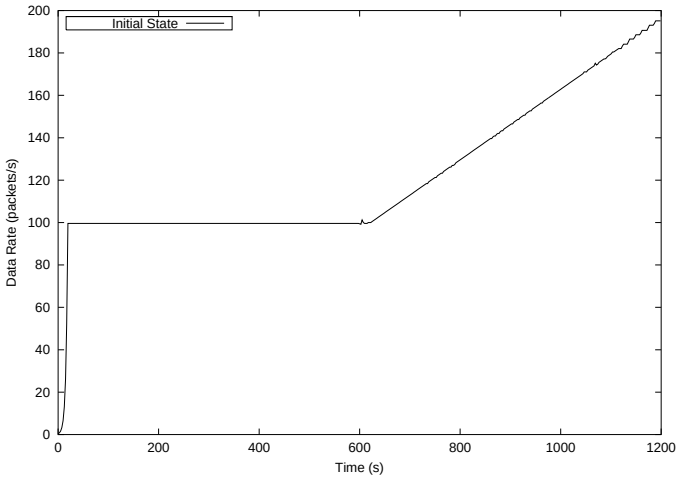

Figure 71: Transmission rate change in the Initial State of TP-Planet source for $RTT = 600$ seconds.

then the ratio of the traffic in the forward and reverse channels is 25:1, i.e., 1KB/40B. Thus, the bandwidth asymmetry up to 25:1 cause no congestion in the reverse link. However, the bandwidth asymmetry in the space links is usually on the order of 1000:1 [44]. Thus, sending one SACK for each data packet can cause reverse channel to be congested resulting in packet losses in the reverse link.

In order to avoid this problem, TP-Planet deploys SACK congestion control by delaying the SACKs. TP-Planet sink maintains delayed-SACK factor, d, and sends one SACK for every d data packets received. If there is no packet loss and hence no change in the SACK blocks, then TP-Planet sink keeps delaying SACKs with a delayed-SACK factor of d. Otherwise, it sends a new SACK with an updated block immediately. Therefore, the amount of traffic on the reverse channel is controlled by adjusting the delayed-SACK factor d. Effects of the blackout on throughput and the improvement achieved by delayed-SACK is evaluated in Section 5.5.4.

5.5 Performance Evaluation

In order to investigate the performance of the TP-Planet, extensive simulation experiments are conducted. The improvement in the initial phase of the connection achieved by the Initial State algorithm is evaluated in Section 5.5.1. Throughput performance of TP-Planet

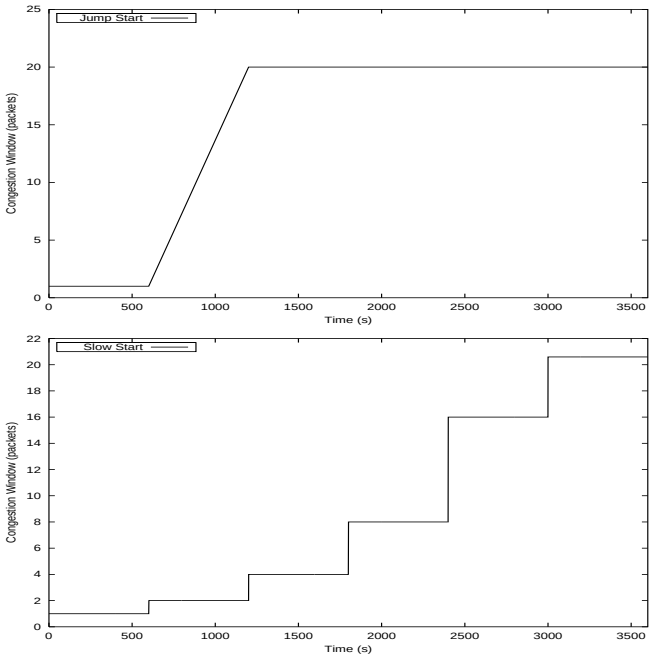

Figure 72: Congestion window size change during Jump Start algorithm of TCP-Peach+and Slow Start algorithm of TCP-NewReno and for $RTT = 600$ seconds.

is analyzed in Section 5.5.2 along with the overhead introduced. Effects of blackout conditions on the performance and the performance of TP-Planet with the improvement by Blackout State are investigated in 5.5.4. TP-Planet performance on deep space links with asymmetrical bandwidth and the improvement with delayed SACK are explored in Section 5.5.5.

5.5.1 Initial State Performance

In order to avoid performance degradation due to inefficient connection starting behavior, TP-Planet deploys `Initial_State()` algorithm in Figure 64 and Figure 67. Here, a topology where the source and sink are connected through a deep space link with $RTT = 600$ seconds and $p = 10^{-5}$ is simulated. The same experiment is performed with TP-Planet, TCP-Peach+ [11] and TCP-NewReno. The reason is that most of the current TCP protocols deploy initial phase behavior based on the Slow Start algorithm, which is also used by TCP-NewReno.

In Figure 71, the transmission rate change in the Initial State of TP-Planet source is plotted, where the target transmission rate of TP-Planet source is assumed to be 100

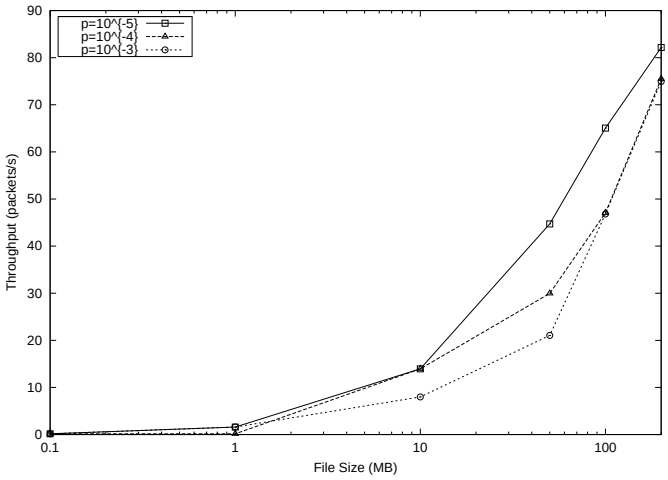

Figure 73: Throughput for changing p and file size with $RTT = 600$ seconds.

packets/s. Because of the very high propagation delay and 100% reliability requirement, the amount of buffer required for retransmission mechanism is proportional to the data transmission rate. Therefore, for very high data rates, this brings considerable amount of memory requirement. For example, the source needs to maintain 0.6 GB of buffer for $RTT = 10$ minutes and the average data transmission rate of 1MB/s. Thus, the target data rate is set to 100 packets/s in order to maintain practicality in terms of memory requirements.

As explained before, TP-Planet source performs Emulated Slow Start phase, which ends once the number of data packets transmitted in one time interval is equal to $ssthresh_e$. As seen in Figure 71, the Emulated Slow Start phase lasts for approximately $T_{ess} = 19.58$ seconds and the TP-Planet source reaches 100 packets/s data rate by then. Emulated Congestion Avoidance phase then starts and lasts until the end of the first RTT of the connection period. During this phase, the data transmission rate is not increased instead NIL segment transmission rate is increased linearly to further probe the link resources. At $t \approx 600$, the source goes to Follow-Up period and increases its transmission rate based on the feedback received from the sink every T period. At $t = 1200$ seconds, the transmission rate reaches 200 packets/s.

TCP-Peach+ [11] deploys *Jump Start* algorithm in order to capture link resources very quickly. In Jump Start, the sender sets the congestion window, *cwnd*, to 1. After sending

the first data segment, it transmits *NIL segments* every $RTT/rwnd$, where $rwnd$ is the receiver window size. As a result, after one round trip time, the congestion window size increases very quickly as the ACKs for NIL segments arrive at the sender. The performance of Jump Start is shown in Figure 72. The congestion window of TCP-Peach+ reaches 20 packets in at most $2 \cdot RTT$ duration. It is, however, still very low compared to the performance of Initial State of TP-Planet, which reaches 200 packets/s data rate at the end of $2 \cdot RTT$.

The slow start performance is, however, not even close to what Initial State of TP-Planet achieves in deep space links of the IPN. In Figure 72, the congestion window evolution dependent on time is shown during the slow start. The slow start period lasts for approximately $6 \cdot RTT$ for threshold window size of 20 packets. Therefore, the link is not efficiently utilized for 60 minutes due to unsuitability of the slow start algorithm to extremely high propagation delays in deep space links. Recall that for higher threshold windows size values, the link is not efficiently utilized for a longer period of time [2].

5.5.2 Throughput Performance

In order to show the throughput performance of TP-Planet in deep space links, several experiments are performed by varying packet loss probability p and the size of the data to be transmitted. The capacity of the link is assumed to be 1Mb/s and $RTT = 600$ seconds. The target transmission rate B is set to be 100 packets/s, i.e., 100 KB/s for data packets of size 1KB. The protocol parameters given in Table 5 are used during simulation experiments unless otherwise stated. The investigation of the effects of these parameters on the protocol performance are left for future study.

In [2], existing TCP protocols including TCP-Vegas, which is adopted by SCPS-TP protocol as a congestion control scheme for space-based communications [38], have been shown to achieve very poor performance in deep space links. For $RTT = 600$ seconds and $p = 10^{-3}$, the throughput achieved by TCP protocols is approximately around 30 B/s and hence the entire link remains almost unutilized [2]. Although TCP-Peach+ has significantly outperformed other TCP schemes for the same link, the performance degradation was still

Table 5: Protocol parameters used in Experiments.

Parameter	Value	Definition
ϕ_i	0.8	Rate increase threshold
ϕ_d	0.2	Rate decrease threshold
τ	5	NIX ACK period in *seconds*
T_w	20	Sliding window size in *seconds*
ξ	0.5	Rate decrease parameter for RTT
d	5	Delayed-SACK factor in *number of packets*

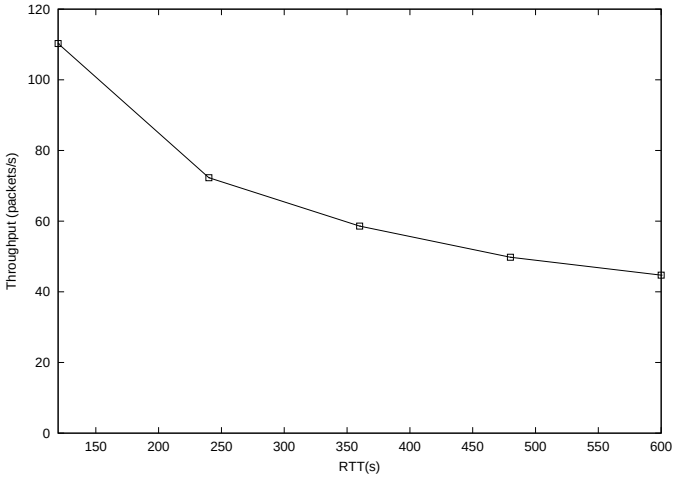

Figure 74: Throughput for changing RTT for file size of 50MB and $p = 10^{-5}$.

too serious that it could achieve throughput around 93 B/s. Thus, the same experiments are not repeated here and the details can be found in [2].

TP-Planet simulation experiments are performed for transmission of varying file sizes between 100KB and 200MB and for varying packet loss probability p of 10^{-5}, 10^{-4}, 10^{-3}. As shown in Figure 73, the throughput increases with increasing file size. This is because the larger the file size is the longer TP-Planet stays in the Steady State. As a result, the link utilization is increased. Although throughput decreases for increasing packet loss probability, this degradation also decreases for increasing file size. This is mostly because throughput improvement by new congestion control scheme is higher when the protocol stays in Steady State for longer. For transmission of 200MB and $p = 10^{-5}$, TP-Planet throughput increases up to 82.2 packets/s approaching its target throughput value, i.e., $B =$ 100 packets/s. Hence, TP-Planet outperforms existing TCP protocols by approximately 10^3

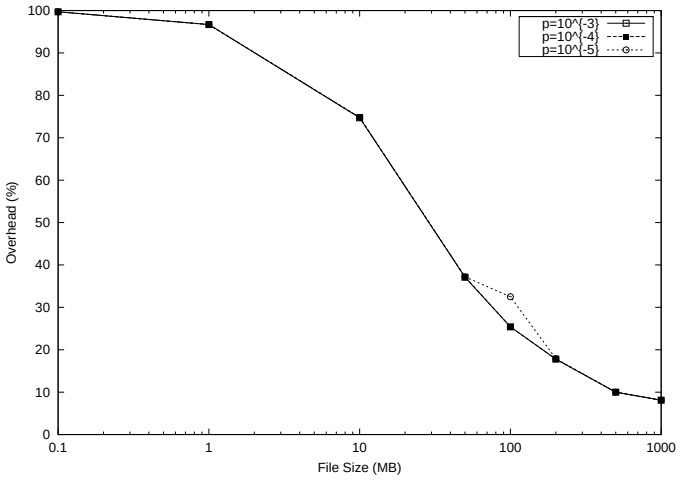

Figure 75: Overhead due to transmission of NIL and NIX segments for changing file size and p with $RTT = 600$ seconds.

times in terms of throughput.

In Figure 74, the effect of RTT on the throughput performance is shown. The experiments are performed for the transmission of a 50MB file. The increase in RTT leads to throughput degradation as shown in Figure 74. However, the throughput degradation due to high propagation delay is not as severe as it is for conventional TCP protocols [2]. This is because of the new rate-based congestion control scheme used in Steady State of the TP-Planet. At $RTT = 600$ seconds, TP-Planet achieves 44.72 KB/s throughput. Note that this value can further increase with increasing file size as in Figure 73.

5.5.3 Overhead

During the connection period, TP-Planet source brings overhead to the deep space link. This overhead is due to NIL segment transmission to probe the link resource in the Initial State and low and high priority NIX packets transmission for congestion decision support in the Steady State. In this subsection, the overhead caused by the injection of NIL and NIX segments into the network is investigated.

In Figure 75, the overhead incurred by TP-Planet is shown for transmission of files with varying sizes and for different packet loss rates, i.e., $p = 10^{-5}, 10^{-4}, 10^{-3}$. For very small files, i.e., $< 10MB$ the overhead is relatively high. Because, in this case, significant portion of the connection period is spent in the Initial State, where the number of NIL segments

transmitted is high compared to that of data packets. As the file size increases, the overhead decreases as in Figure 75. This is because the overhead in the Steady State is due to the transmission of small sized (40 bytes) NIX segments and hence is much lower compared to Initial State. Therefore, as the file size increases, the time spent in Steady State also increases, which in turn decreases the overall overhead in the connection period. As a matter of fact, the scientific data delivered during space exploration missions are significantly high [22], i.e., in the order of gigabytes, which leads to very low overhead.

On the other hand, the overhead does not significantly vary for different p. As a matter of fact, the amount of NIL segments transmitted in the Initial State depends only on the target throughput and the RTT and thus it is not dependent on p. This overhead exists only in the beginning of the connection. Although the NIL segment transmission during Initial State brings overhead, it also leads to significant performance improvement as observed in Section 5.5.1. As the NIX transmission rate is equal to data rate and the congestion control is robust to link errors, the overhead due to NIX transmission is also independent of p. The overhead due to NIX transmission exists after the Initial State until the end of the connection. However, the congestion control decision support provided by NIX segments lead to significant throughput performance improvement. For 1GB file size and $p = 10^{-3}$, the overhead becomes as low as 8.1 %, which is quite low compared to throughput improvement with a factor more than 10^3.

5.5.4 Blackout Conditions

When a blackout is detected, TP-Planet moves to Blackout State as shown in Figure 64 as explained in Section 5.4.3. Throughput achieved by TP-Planet for different blackout durations is given in Figure 76. Here, $RTT = 120$ seconds, $p = 10^{-5}$ and the target data rate is assumed to be 50 KB/s. The simulations are performed for a duration of 600 seconds, where the blackout occurs at $t = 250$ seconds.

As in Figure 76, throughput decreases with increasing blackout duration as expected. This decrease is observed in both of the curves representing the TP-Planet operation with and without Blackout State procedure. However, Blackout State procedure improves the

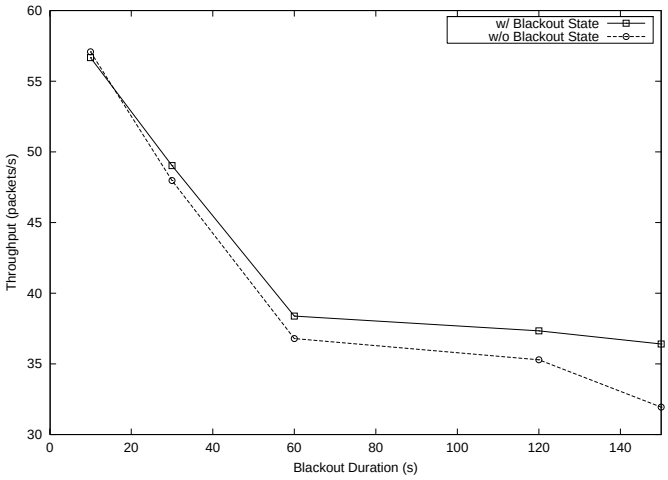

Figure 76: Throughput for changing blackout duration with $RTT = 120$ seconds and $p = 10^{-5}$.

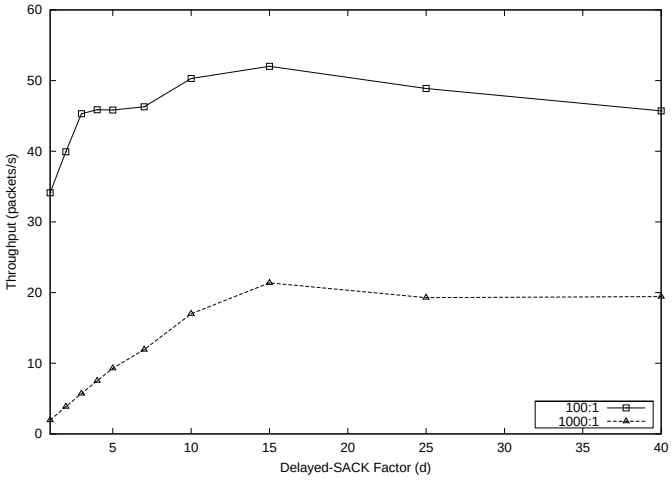

Figure 77: Throughput for changing delayed SACK factor d with $RTT = 120$ seconds and $p = 10^{-4}$.

performance for long blackout durations. For even a blackout of 150 seconds, which is 1/4 of the entire simulation time, TP-Planet achieves 36.41 packets/s throughput with the help of Blackout State behavior. This corresponds to approximately 14% performance improvement over the case without Blackout procedure.

5.5.5 Bandwidth Asymmetry

In order to address bandwidth asymmetry problems, TP-Planet delays SACKs with a certain delayed SACK factor. Simulation experiments are performed to show the effect of

bandwidth asymmetry on the performance and the improvement achieved by delayed SACK. Here, $RTT = 120$ seconds $p = 10^{-4}$ and the simulation time is assumed to be 1200 seconds. The target rate is set to 50 KB/s. In Figure 77, two cases with different bandwidth asymmetry ratio are investigated:

1. **100:1.** In this case, forward and reverse link capacities are 100 KB/s and 1 KB/s, respectively. Since the ratio of the size of data packets and SACK packets is 25:1, the actual asymmetry ratio is 4:1 in this case. Therefore, the throughput is not significantly degraded by asymmetrical channel capacity of the deep space link. As shown in Figure 77, an increase in the delayed SACK factor also leads to an increase in the throughput. However, throughput decreases for both cases with either too high or too small delayed SACK factor of d. This is because if it is too small, reverse channel is congested. On the other hand, if d is too large, the source receives less number of SACKs than it expects which leads to performance degradation. Thus, at $d = 15$, throughput achieved increases up to 52.02 packets/s.

2. **1000:1.** In this experiment, bandwidth asymmetry on the order of 1000:1 is simulated by setting forward and reverse link capacities to 100 KB/s and 0.1 KB/s, respectively. Thus, the throughput is dramatically affected by the bandwidth asymmetry in this case, where the actual bandwidth asymmetry is 40:1. Therefore, throughput improvement with delayed SACK method is more significant for higher bandwidth asymmetry. For the case without delayed SACK, throughput decreases to up to 1.99 KB/s. As in Figure 77, throughput increases with increasing number of delayed SACKs. However, this pattern does not last long since very high delayed SACK factors cannot be reached. This is because a data packet loss yields new SACK blocks, which requires to be transmitted back to the source immediately. For delayed SACK factor of 15, throughput increases up to 21.37 KB/s which corresponds to 10 times increase in the throughput performance compared to the case without delayed SACK, i.e., $d = 1$ in Figure 77.

CHAPTER VI

INTEGRATED TRANSMISSION PROTOCOL FOR INTERPLANETARY INTERNET

In this chapter, a new Integrated Transmission Protocol (ITP) is presented for reliable data transport in the IPN Internet. ITP is a unified transmission protocol solution for hop-by-hop congestion control and reliability mechanisms specifically tailored for IPN Internet links with intermittent connectivity where end-to-end conventional transport layer mechanisms are inherently inefficient. ITP was first presented in [9]. A comprehensive review of the related work on reliable data transport in the IPN Internet is presented in Section 6.2. Then, the bundling approach is explained and the case for the hop-by-hop integrated transmission protocol solution is introduced in Section 6.3. The ITP protocol overview along with the detailed operations of the protocol algorithms are explained in Section 6.4. The performance evaluation of ITP and the simulation results are presented in Section 6.5.

6.1 Motivation

As described in Chapter 5, the InterPlaNetary (IPN) Internet is envisioned to be composed of heterogeneous architectural elements such as IPN Backbone Network, IPN External Networks, and PlaNetary Networks. The main characteristics of IPN paths have been outlined as (i) very long propagation delays, (ii) high link error rates, (iii) intermittent connectivity, and (iii) bandwidth asymmetry. These challenges and their consequent effects need to be addressed in order to meet the communication requirements of deep space missions and realize the IPN Internet. However, in [2], the existing reliable transport protocols have been shown to achieve very poor performance in deep space communication networks mainly because of the inefficiency caused by their end-to-end window-based congestion control mechanisms due to very high propagation delay and link errors.

In general, the end-to-end congestion control and reliability mechanisms are deemed

to suffer from the adverse effects of the above challenges. The conventional transport layer approaches with end-to-end congestion control and reliability mechanisms assume the existence of an end-to-end path from the source to the final destination. Such assumption may not be totally unreasonable for some IPN Internet paths with very few hops such as the Earth-Mars communication network despite occasional blackout situations. In fact, a novel end-to-end reliable data transport protocol, TP-Planet [6], is shown to efficiently address the above challenges and significantly improve the throughput performance in the Earth-Mars communication architecture. However, intermittent connectivity is one of the main characteristics of the IPN Internet paths and hence should be considered as part of any connection period rather than an occasional and extreme network condition. In this case, end-to-end transport layer protocols experience significant network inefficiency due to frequent retransmissions performed all the way from the source to the destination. This problem is further amplified in the IPN Backbone Network paths spanning several IPN hops, which increases the probability of experiencing blackouts in a given connection period. Moreover, due to extremely high propagation delay, end-to-end congestion control mechanisms are severely affected by the delayed and even obsolete network or receiver feedback and hence they lose accuracy in congestion decisions.

Consequently, the hop-by-hop transport approach based on link-level local flow control and reliability mechanisms become a viable alternative to inefficient end-to-end approaches for the IPN Backbone Network paths. In this direction, the Delay-Tolerant Networking Research Group (DTNRG) [108] introduces the *bundling* approach [95], which performs a *custody-based store-and-forward* transport of the data *bundles*, i.e., package of messages. This approach is based on the *tiered Automatic Repeat reQuest (ARQ) reliability* and *tiered congestion control* concepts [30] to provide reliable transmission via local retransmissions and perform congestion control on the regional basis, i.e., locally between bundling nodes instead of end-to-end data transport reliability and congestion control, while the reliable transport of a bundle from the originating application to the receiver application is assured by the bundle layer reliability [33].

In the hop-by-hop approach, the congestion control and reliability mechanisms are performed on a local link-level basis to achieve overall end-to-end reliable transport. This also brings a significant potential advantage of integrating transport and link layer functionalities that are in common such as transmission flow control and packet level reliability. However, there is no such solution which exploits this potential advantage and provides reliable data transport for custodial store-and-forward bundling approach in the IPN Internet. All these above discussed challenges and the inherent inefficiencies of the end-to-end approaches call for a unified, efficient, and integrated hop-by-hop transmission protocol for reliable data transport in the IPN Internet. In order to address this need, a new Integrated Transmission Protocol (ITP) for the IPN Internet is presented in this chapter.

ITP is a new *integrated transmission protocol including hop-by-hop local flow control and reliability mechanisms developed to address the challenges and to achieve high performance reliable data transmission over deep space links of the IPN Backbone Network.* Some of its salient features are as follows:

1. *New Local Flow Control Mechanism:* ITP deploys a new rate-based hop-by-hop local flow control (LFC) mechanism; which exploits the local resource availability and traffic information at the receiver in order to provide explicit available bandwidth feedback to the sender. LFC mechanism decouples congestion decision from reliability to avoid the erroneous congestion decisions due to high link errors and avoids the delayed-feedback problem as described in Section 6.4.1.

2. *Local Packet-Level Reliability:* ITP is a hop-by-hop integrated transmission protocol which avoids the need for both transport layer and link layer reliability mechanisms by assuring local link-level reliability. ITP incorporates the selective-acknowledgment (SACK) based Automatic Repeat reQuest (ARQ) to assure hop-by-hop packet-level reliability as described in Section 6.4.2.

3. *Blackout Mitigation Procedure:* In order to reduce the effects of blackout conditions on the throughput performance, ITP incorporates Blackout Mitigation (BM) procedure into the protocol operation. The objective of BM procedure is to accurately

capture the blackout information so as to minimize the resultant degradation in the link utilization as explained in Section 6.4.3.

4. *Optimum Packet Size:* The extreme propagation delay decreases the link efficiency, which can be increased by higher packet size. However, higher packet size also increases the packet loss rate for a given bit error rate (BER). Consequently, the increase in packet loss probability further decreases the link utilization by increasing the amount of bandwidth used for retransmissions performed by the ARQ error control. Based on these constraints, in Section 6.4.4, the *optimum packet size* is determined to be used by the ITP protocol in order to maximize the link efficiency.

5. *Delayed SACK:* Bandwidth asymmetry problem is addressed by the adoption of delayed selective-acknowledgments (SACK), which reduces the traffic in the reverse channel to avoid congestion in the reverse channel as described in Section 6.4.5.

6.2 Related Work

The challenges posed by the IPN Backbone links need to be addressed in order to meet the communication requirements of deep space missions and to realize the IPN Internet. In Section 5.2, the existing transport layer mechanisms have been throughly investigated and it has been concluded that that the existing schemes are far from addressing the challenges imposed by the IPN Internet.

In Chapter 5, a reliable transport protocol, TP-Planet, for IPN Backbone Network has been introduced to address the challenges involved in reliable data transport over IPN Backbone Network links. It has been shown in [6] via simulation experiments that TP-Planet significantly improves the throughput performance and addresses the challenges in IPN Backbone Network for the Earth-Mars communication architecture. Nevertheless, TP-Planet is an end-to-end solution which is subject to adverse effects of high propagation delay and intermittent connectivity such as inadequate or inaccurate congestion decisions and control actions due to delayed-feedback and blackout situations. Furthermore, TP-Planet is developed mainly for Mars exploration mission communication architecture. Hence, it

may experience further severe performance degradation as the IPN Backbone Network path increases to the farther planets such as Jupiter and Pluto.

In [95], the *bundling* protocol is introduced to address the intermittent connectivity, large and variable delays, and high bit error rates. As shown in Figure 82, the bundling protocol sits between the application and the lower layers, and performs a custody-based *store-and-forward* approach for the *delay-tolerant networking* (DTN) architecture introduced in [45]. Based on the store-and-forward functionality of the bundling layer, the DTN approach incorporates *tiered ARQ* and *tiered congestion control* concepts [30] to provide reliable transmission via local retransmissions and perform congestion control on the regional basis, i.e., locally between bundling nodes instead of end-to-end reliability and congestion control. Although this approach achieves reliable transport over intermittent links, it still requires a new reliable data transport mechanism, which is specifically tailored to address the same challenges, to achieve high throughput performance in hop-by-hop bundle transport between two IPN Internet nodes.

Consequently, despite the existence of a certain amount of research on reliable data transport in deep space networks, there is no solution which exploits the potential advantage of hop-by-hop custodial transport paradigm and provides high performance reliable data transport for custodial store-and-forward bundling approach in the IPN Internet.

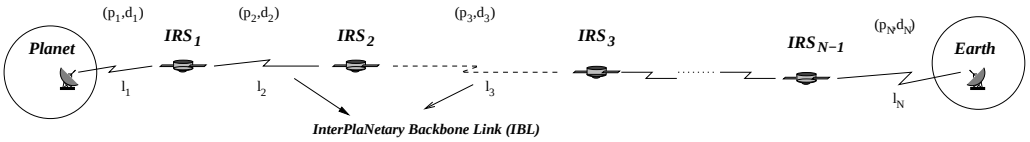

Figure 78: An illustration of a complete IPN Internet path and the hop-by-hop and the end-to-end transport approaches in case of blackout.

6.3 Custodial Transport and Case for Integrated Transmission Protocol

In the hop-by-hop bundling paradigm introduced in [30], the transport of a data bundle is performed in a custodial store-and-forward approach. In this case, a bundle transmission starts at one IPN Internet node and ends at the next one. The sink successfully receiving the

entire bundle, then starts forwarding to the next IPN Internet node. Hence, at a given time, the connection takes place on a single IPN link, where the congestion control and reliability mechanisms are performed on a local link-level basis to achieve overall end-to-end reliable transport of the bundle originated at the source application to the final destination node.

The main objective of such hop-by-hop store-and-forward approach is to address the significant challenge of intermittent connectivity experienced in the IPN Internet. In fact, unlike the traditional wireless and satellite networks, blackout conditions should be considered part of normal IPN experience for any connection rather than an occasional and extreme network condition. Hence, considering the significance of the data gathered in scientific deep space missions, it is wiser to exploit any present intermittent contact opportunity to the fullest instead of waiting for the availability of an end-to-end path throughout the IPN Internet.

As an example, consider the following scenario illustrated in Figure 76. Here, a typical communication path between the Earth to an outer-space planet consists of several IPN hops, i.e., IPN Relay Stations (IRS). The scientific data gathered at the outer-space planet is being transfered to the Earth ground station over this path. Assume that the link between IRS_2 and IRS_3, i.e., l_3, is broken during the connection period due to a blackout of an unknown duration. In the end-to-end case, the entire transport communication would halt until l_3 is back. Moreover, inefficient end-to-end retransmissions would be mandatory all the way from the planet to the Earth in order to recover from packet losses due to blackout and hence to assure the reliability of the data transport. However, in the hop-by-hop custodial approach, the data could be transported up to IRS_3 and can be stored there until the blackout is over. Even in the worst case that the blackout occurs while the bundle is being transported from IRS_3 to IRS_2, its effect would be much less than the end-to-end scenario and only local retransmissions would be required to assure reliability.

In order to further understand the differences between the end-to-end and hop-by-hop custodial approaches, the transmission efficiencies of each approach are analytically investigated. Here, the network configuration shown in Figure 76 is considered, where exist $N-1$ hops, i.e., $IRSs$, and hence N IPN links, i.e., l_i is the link between IRS_{i-1} and IRS_i for

$i \leq N$, between the outer-space planet and the Earth. Here, the transmission efficiencies of the end-to-end ($e2e$) transport and hop-by-hop (hbh) approaches are of interest. For this purpose, the expected total number of packets transmitted from the planet to the Earth is calculated including the retransmissions in order to reliably transport a bundle of size B using data packets of size L. Let $E[\mathcal{N}_{e2e}]$ and $E[\mathcal{N}_{hbh}]$ be the expected total number of packets transmitted in $e2e$ and hbh approaches, respectively. Then, $E[\mathcal{N}_{e2e}]$ and $E[\mathcal{N}_{hbh}]$ can be expressed as [9]

$$E[\mathcal{N}_{e2e}] = \sum_{k=0}^{\infty} \left\{ D \sum_{i_1=1}^{N} \sum_{i_2=1}^{N} \dots \sum_{i_k=1}^{N} p_{i_1} p_{i_2} \dots p_{i_k} \times \left(\prod_{i=1}^{N} (1 - p_i) \right) \left[N + \sum_{l=1}^{k} i_l \right] \right\} \quad (68)$$

and,

$$E[\mathcal{N}_{hbh}] = \sum_{j=1}^{N} \sum_{i=0}^{\infty} D(1 - p_j) p_j^i (i + 1) \quad (69)$$

where N is the number of links l_i for $i = 1..N$; p_i is the packet loss probability on the link l_i; B is the size of the total data, i.e., bundle size; L is the packet size and hence D is the number of packets generated for a bundle of size B, i.e., $D = B/L$.

In order to be able to compare the performances of the end-to-end ($e2e$) and hop-by-hop (hbh) transport approaches, $E[\mathcal{N}_{e2e}]$ and $E[\mathcal{N}_{hbh}]$ are plotted for B, N, and p values as shown in Figure 80, 81 and 79, respectively. As it is observed from the plots, the expected total number of packet transmissions is always higher in the $e2e$ approach than the hbh case, i.e., $E[\mathcal{N}_{e2e}] > E[\mathcal{N}_{hbh}]$. More specifically, as shown in Figure 80, the difference significantly increases as the number of hop count increases. Similar behavior is also observed as the packet loss probability and bundle size (hence the total number of packets generated for a given bundle, i.e., D) increases. This is because the number of retransmissions increases with experiencing more packet losses due to increasing packet loss probability and more blackouts as the end-to-end path involves higher number of intermittent IPN links. Consequently, the hop-by-hop approach is indeed more transmission efficient for the reliable data transport in the IPN Internet characterized by intermittent connectivity and high link errors.

In addition to its greater transmission efficiency as observed in Figure 79, 80, and 81; such hop-by-hop transport approach also brings a significant potential advantage of integrating transport and link layer functionalities that are in common such as transmission flow

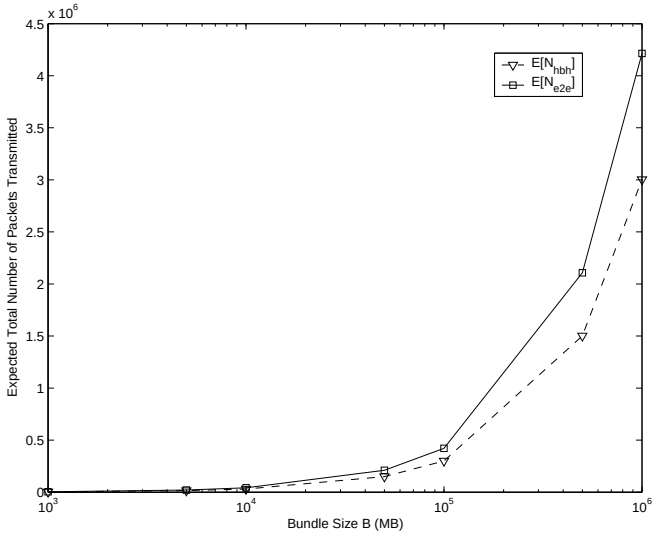

Figure 79: Comparison of $E[\mathcal{N}_{e2e}]$ and $E[\mathcal{N}_{hbh}]$ for varying bundle size B.

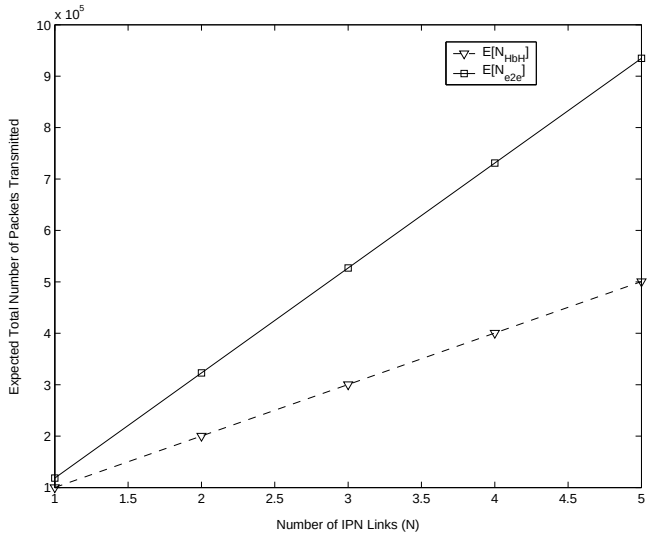

Figure 80: Comparison of $E[\mathcal{N}_{e2e}]$ and $E[\mathcal{N}_{hbh}]$ for varying number of IPN links N.

control and packet level reliability. As the transport layer congestion control and reliability mechanisms are for end-to-end purposes, in this case, there is no need for separate transport and link layer flow control and reliability mechanisms. Instead, a unified integrated flow control and reliability mechanism is more efficient approach for hop-by-hop store-and-forward bundling paradigm. Hence, there exists a need for such an integrated transmission protocol solution which includes efficient flow control and reliability mechanisms to achieve high performance reliable data transport in the IPN Internet. In the next section, the

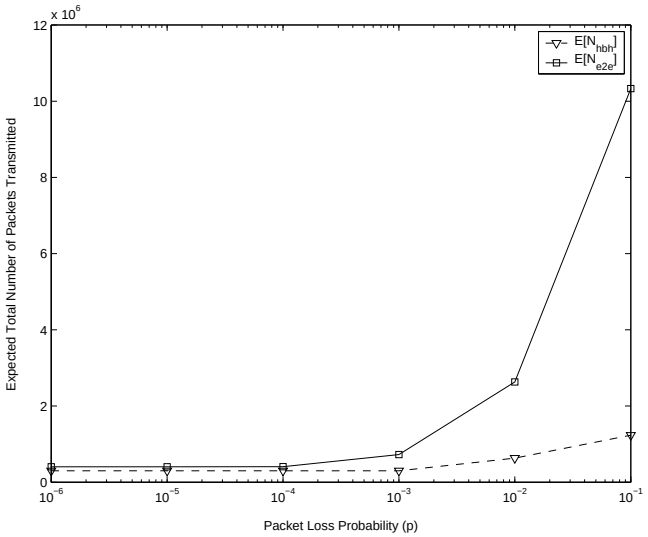

Figure 81: Comparison of $E[\mathcal{N}_{e2e}]$ and $E[\mathcal{N}_{hbh}]$ for varying packet loss probability p.

Integrated Transmission Protocol (ITP) is developed to address this need.

6.4 ITP: Integrated Transmission Protocol

As discussed in Section 6.3, the hop-by-hop bundling approach brings the possible potential of integrating transport and link layer flow control and reliability mechanisms in order to perform local flow and packet-level error control in the IPN Internet. In order to exploit this and achieve high performance reliable data transport, the Integrated Transmission Protocol (ITP) is developed for IPN Internet. ITP is a new integrated transmission protocol which incorporates hop-by-hop local flow control and packet-level reliability mechanisms specifically developed to address the challenges and to achieve high performance reliable data transmission over deep space links of the IPN Internet. ITP can be implemented at the IPN nodes such as the IPN gateways and IPN relay satellites in between the planets and the Earth as part of the IPN Backbone Network as well as at the IPN relay satellites/spacecrafts orbiting around the outer-space planets as part of the PlaNetary Networks as shown in Figure 2.

The protocol layer structure of the ITP is illustrated in Figure 82. ITP can work in cooperation with the *bundling layer* developed by the DTNRG [33] in order to provide reliable data transport functionalities for custodial store-and-forward bundling transport

as shown in Figure 82. It replaces the conventional transport and link layers except the medium access control (MAC) and framing functionalities which are beyond the scope of this work. In the next subsections, the ITP protocol algorithms are presented in detail and the optimum packet size to be used by the ITP is determined in order to maximize the transmission efficiency over the deep space links of the IPN Internet.

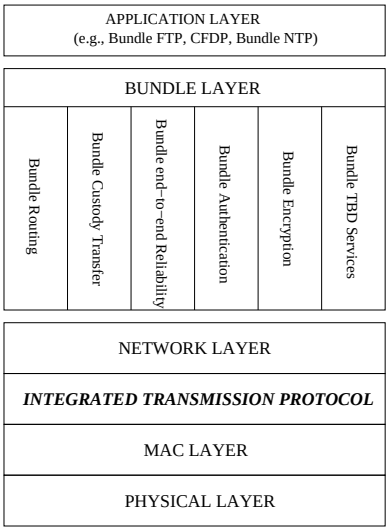

Figure 82: A typical IPN protocol stack including the ITP protocol in coordination with the bundling layer.

6.4.1 Local Flow Control (LFC) Mechanism

As discussed in Section 6.3, the bundling approach incorporates the hop-by-hop custodial store-and-forward transport of data bundles between IPN Internet nodes. A typical bundle transport between two IPN relay satellites is illustrated in Figure 83. Here, the steps of the bundle transport can be summarized as follows:

- The transport of a *bundle* i starts at the sender IPN relay station, i.e., IRS_s, and ends at the receiver IRS, i.e., IRS_r, which is the designated next hop to the sender.
- IRS_r, successfully receives the entire bundle, acknowledges the successful reception to its sender, i.e., IRS_s; and it only then starts transmitting to the next hop.

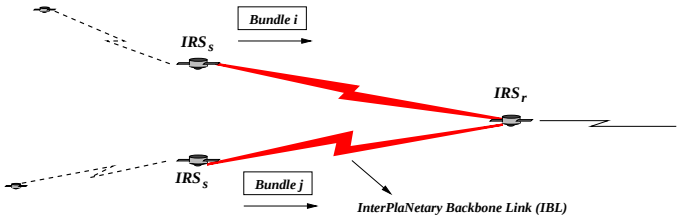

Figure 83: An illustration of a typical bundle transport between IPN relay satellites.

Note that, in this scenario, an actual communication for a bundle transport takes place on a single link between two IPN nodes at a given time. The traditional congestion control mechanisms, however, are not suitable in such custodial transport approach. This is mainly because the congestion condition itself is different in this scenario. In the traditional routing/forwarding behavior, congestion mainly occurs at the queue buffers of a router due to competing sources for the router processing and the output link resources. However, in hop-by-hop scenario, an IPN sender/forwarder node stores a bundle and then starts to forward after it is completely received rather than directly forwarding whatever it receives. Hence, the congestion can occur at the receiver IPN node, IRS_r, due to the simultaneous bundle transmissions competing for the same input sources, i.e., ingress link bandwidth capacity and bundle storage buffer.

In addition to this, the congestion detection and control methods based on active link probing such as additive-increase multiplicative-decrease (AIMD) mechanisms adopted by the TCP protocols have a major problem of delayed feedback in the IPN Internet links with very high propagation delays. In such approach, a congestion condition may already be resolved at the time it is just detected by the source and hence the action taken may lead to inaccurate and undesirable results. Therefore, local flow control mechanism which is independent of the adverse effects of the propagation delay is essential to perform accurate and efficient congestion detection and flow control in very high delay IPN Internet links.

ITP protocol incorporates a new local flow control (LFC) algorithm to address the above challenges and achieve high link utilization in reliable bundle transport between the IPN Internet nodes. LFC algorithm decouples congestion control and reliability mechanisms and does not use packet-loss based congestion detection method. The main principle of the LFC

algorithm is that the sender performs a new rate-based flow control and explicitly sets the data rate R_k for transport of the bundle k utilizing the explicit feedback provided by the receiver based on the link capacity and bundle storage buffer availability.

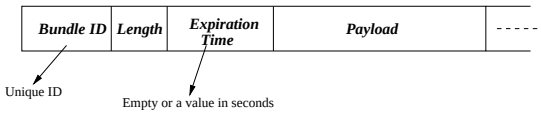

Figure 84: A typical simplified bundle structure defined by the DTNRG with some of the bundle header fields.

The ITP protocol operation starts as the ITP layer receives the bundle transport request from the bundling layer residing at the sender node. ITP source divides the entire bundle into data packets whose length is optimized as described in Section 6.4.4. A typical simplified bundle structure defined by in the Bundling Specifications [95] with some of the bundle header fields is shown in Figure 84. Each bundle k has certain length, i.e., S_k, and may have a time constraint, i.e., expiration time δ_k, after which the data contained in the bundle is obsolete based on the mission/application requirements [95]. Each data packet generated for bundle k carries its bundle-specific information in its header, e.g., unique bundle ID, length S_k and expiration time δ_k.

In the LFC operation, ITP receiver performs local flow control decision, calculates and sends back to the ITP sender the amount of link capacity it grants to the requested bundle transport using its local resource availability information, i.e., ingress link bandwidth capacity and bundle storage buffer. For this purpose, each IPN relay station (IRS) monitors certain local information as also shown in Figure 85 with the following terminology:

N_i number of bundles it currently receives,

N_o number of bundles it currently sends,

S_k length of the *bundle* k,

$W_k(t)$ size of the received and stored portion of the *bundle* k at time t,

$R_k(t)$ link utilization (instantaneous data rate) of transport of bundle k at time t,

δ_k the expected time delivery value of the *bundle k*.

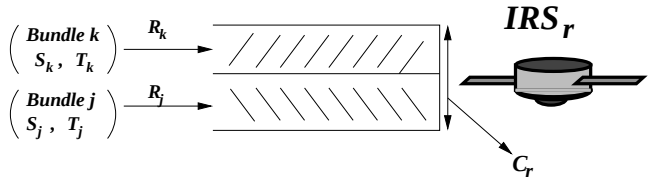

Figure 85: Illustration of two simultaneous bundle transport to an IRS monitoring its local information.

At the beginning of the connection to transport the bundle k, the sender IRS, IRS_s, first checks if the link is up and there is available bandwidth at its output port towards IRS_r, i.e., the next hop for the bundle to be transported. Let C_{sr} be the total output bandwidth capacity at the IRS_s. Then, the transport request for the bundle k is granted if the link is up and there exists available bandwidth, i.e., $C_{sr} > \sum_j^{N_o} R_j$, where N_o is the total number of bundles and R_j is the link utilization of bundle j currently being sent by IRS_s for $j = 1..N_o$. Otherwise, the bundle k is kept in the bundle storage buffer until the link is recovered and bandwidth is available.

ITP source, IRS_s, starts the connection for the granted transport request of the bundle k by sending connection request message. If the bundle does not have any time constraint specified by the application in the bundle header, i.e., $\delta_k = 0$, then connection resumes after the ACK for the connection request is received. If the bundle has a certain time constraint, i.e., $\delta_k \neq 0$, then IRS_s directly starts sending data packets at $t = 0$ using the initial transmission rate $R_k(0)$ determined as

$$R_k(0) = \min\left\{\left[C_{sr} - \sum_k^{N_o} R_j\right], \hat{R}_k\right\} \tag{70}$$

where C_{sr} is the output capacity of the IRS_s, R_j is the current transmission rate for the bundle j, N_o is the total number of bundles being transported by IRS_s, and $\hat{R}_k$ is the expected average link share for the transport of the bundle k to meet its time constraint δ_k, i.e., $\hat{R}_k = S_k/\delta_k$.

Let RTT be the round-trip time between the sender and receiver, i.e., IRS_s and IRS_r. At $t = RTT/2$, the IRS_r receives the bundle-specific information carried in the packet

headers for the bundle k, i.e., bundle ID, length S_k, and δ_k. It then uses this bundle information and its local resource availability information, i.e., input link bandwidth capacity and bundle storage buffer, in order to decide whether or not it can handle the incoming bundle transmission and to calculate the bandwidth capacity it can allocate as described follows.

Let N_i^r be the total number of active incoming bundle transmissions at the IRS_r, and S_j and $W_j(t)$ be the total length and the size of the received portion of the bundle j at time t for $j = 1..N_i^r$. The objective is to find the amount of available bandwidth IRS_r can allocate for the transport of bundle k, i.e., R_k. ITP protocol considers both the available bandwidth capacity and the buffer availability information while calculating the bandwidth to be allocated for a bundle transport request as described following.

Let R_j be the current data rate of an active incoming transmission of bundle j for $j = 1..N_i^r$ and C_r be the input link capacity of the IRS_r. Let C_i be the total incoming traffic amount IRS_r can accommodate to avoid any buffer overflow. Note that this total incoming link utilization, C_i, depends also on the buffer utilization at the IRS_r to assure reliable storage for active bundle transports and avoid any buffer overflow and therefore $C_i \leq C_r$.

Let the bundle transport request be received at the IRS_r at $t = t_0$ and $B_A(t)$ be the instantaneous amount of available bundle storage buffer at the IRS_r at time t. Hence, $B_A(t)$ can be expressed as

$$B_A(t) \quad = \quad B_r - B_u(t_0) - C_i(t - t_0) + \int_{t_0}^{t} C_o(\delta)\mathrm{d}\delta \tag{71}$$

where B_r is the total buffer capacity, $B_u(t)$ is the amount of used buffer resources at time t, C_i is the total input traffic amount IRS_r can accommodate, C_o is the instantaneous total output link utilization. Hence, to avoid any buffer overflow, $B_A(t) > 0$ must hold during the connection period of the currently active and the requested bundle transports. Therefore, it follows from (71) that the total available input bandwidth that IRS_r can accommodate C_i achieving maximum utilization of link capacity without leading to buffer overflow can

be calculated by

$$C_i = \min\left\{C_r, \frac{1}{t_{max} - t_0}\left[B_r - B_u(t_0) + \int_{t_0}^{t_{max}} C_o(\delta)\mathrm{d}\delta\right]\right\} \tag{72}$$

where t_{max} is the expected remaining transport duration of the bundle k where $k = \arg\max\{S_k - W_k\}$.

LFC algorithm provides fair share of available bandwidth resources, i.e., equally distributes the total incoming link resources IRS_r can accommodate, C_i, among the bundle transports, i.e., $R_j = C_i/N_i^r$ for $j = 1..N_i^r$. Hence, the data rate which can be initially assigned to the bundle k by the IRS_r is $R_k = C_i/(N_i^r + 1)$ due to this equal bandwidth share discipline. However, since some of the active incoming bundle transmissions may end while the bundle k is being received, this data rate R_k to be allocated to bundle k is only valid for a certain period of time. Hence, R_k needs to be updated as the available link resources at the IRS_r increases in order to increase the link utilization. Therefore, since at the time of the request there are N_i^r active incoming bundle transmissions at the IRS_r, the data rate determined for bundle k, i.e., R_k, may need to be updated at most N_i^r times, i.e., for the bundle transport which are currently active but ends while bundle k is being transported. Thus, IRS_r calculates the remaining connection durations of each of the incoming bundle transmissions, i.e., τ_j for $j = 1..N_i^r$. Consequently, the ITP receiver calculates the data transmission rates it can accept for bundle k, i.e., R_k^j, and the time durations that these rates are valid, i.e., τ_j, and send this *availability timeline information* (ATI) pairs, i.e., (R_k^j : τ_j) back to the ITP sender.

Assume that τ_j be ordered using the $S_j - W_j$ information for $j = 1..N_i^r$ in the increasing order, i.e., $\tau_1 < \tau_2 < \ldots < \tau_{N_i^r}$. Then, τ_1 can be expressed by

$$\tau_1 = \frac{S_1 - W_1 - R_1\frac{RTT}{2}}{R_1\left(\frac{N_i^r}{N_i^r+1}\right)} \tag{73}$$

where S_1 is the total size; W_1 is the size of the received amount at the IRS_r; and R_1 is the current data rate of bundle 1.

Since the data rate of a currently active bundle transport is also updated as any other bundle transport finishes, the remaining transmission durations of bundles $j \geq 2$ is affected

```
LFC()
 For bundle k;
  Sender IPN Node:
    If (link is up and C_sr > sum_j^{N_o} R_j)
      Send CONN_REQUEST;
      Set R_k(0) by (70);
      /* If bundle k time constrained */
      If (δ_k = 0)
        Wait for CONN_ACK;
      end;
      Start transmission with R_k(0);
    end;
    If (ATI_RECEIVED)
      /* Explicit rate control with LFC algorithm */
      /* using (R_k^j : τ_j) pairs */
      Set R_k = R_k^j for RTT < t ≤ RTT + τ_j;
      Send DATA packets with rate R_k;
    end;
    If (NO_ACK in T_w)
      /* Blackout Mitigation (BM) Procedure */
      Stop sending new packets;
      If ((BM_ACK) || (DATA_ACK) ARRIVED)
        /* (L ≥ 2x) || (L < 2x) */
        Retransmit TIMEOUT packets;
        Resume LFC operation;
      end;
    end;
  Receiver IPN Node:
    Calculate input capacity C_i by (72);
    Sort τ_j with (S_j − W_j) for j = 1..N_i^r;
    Calculate τ_j by (73) and (74);
    Calculate R_k^j by (75);
    Send (R_k^j : τ_j) (ATI pairs) to source;
    If (Change in C_i or τ_j for j = 1..N_i^r)
      Recalculate R_k^j and τ_j;
      Resend (R_k^j : τ_j) (ATI pairs);
    end;
 end;
end;
```

Figure 86: A pseudo-algorithm for the operation of the `LFC()` algorithm.

by this variation in their corresponding data rates R_j for $j \geq 2$. Hence, following the similar logic used in deriving (73), τ_j for $j \geq 2$ can be calculated by

$$\tau_j = \frac{S_j - W_j - R_j \left(\frac{RTT}{2} + \frac{N_i^r}{N_i^r+1}\tau_1 + \ldots + \frac{N_i^r}{N_i^r-j+3}\tau_{j-1} \right)}{R_j \left(\frac{N_i^r}{N_i^r-j+2} \right)}$$

using (73) and iteratively solving above follows

$$\tau_j \quad = \quad \frac{S_j - W_j - (S_{j-1} - W_{j-1})}{R_j \left(\frac{N_i^r}{N_i^r-j+2} \right)} \tag{74}$$

where S_j is the size; W_j is the size of the received amount of bundle j at the IRS_r; R_j is the current data rate of bundle j; and N_i^r is the total number of active incoming bundle transmissions at the IRS_r.

As the data rate for the transport of bundle k will be updated up to N_i^r times, there will be up to N_i^r ATI pairs which include data rate R_k^j to be used for a duration of τ_j for $j < N_i^r$. Therefore, according to the equal bandwidth share discipline, the data rate for the transport of bundle k, R_k^j, for a duration of τ_j for $j < N_i^r$ can be expressed by

$$R_k^j = \frac{C_i N_i^r}{(N_i^r - j + 2)} \tag{75}$$

where N_i^r is the total number of incoming bundle transports and C_i is the available input bandwidth capacity of the IRS_r as in (72).

After the ATI pairs for a transport request of the bundle k, i.e., $(R_k^j : \tau_j)$ for $j = 1..N_i^r$, are determined using (73), (74), and (75); the ITP receiver grants the transport request for bundle k, and transmits these ATI pairs back to the ITP source, i.e., IRS_s. Note that these ATI pairs are transmitted within data ACK packets back to the ITP source.

The ITP source IRS_s receiving these ATI pairs at $t \approx RTT$, updates the transmission rate for the transport of bundle k using the received explicit data rates R_k^j and their validity durations τ_j. It also sets the timer variable t_{next_update} to schedule the next data rate update using the validity duration, i.e., $t_{next\_update} = RTT + \tau_1$. Hence, the data rate for transport of bundle k is explicitly set by using the receiver-based information carried in the ATI pairs, i.e., $R_k = R_k^j$ for $RTT < t \leq RTT + \tau_j$. The overall LFC algorithm operation is summarized in Figure 86.

On the other hand, if there is any new request for the IRS_r and the link availability information changes and then this change is reported to all of the IRS_s's which are currently transporting bundles to IRS_r. Note that unlike the Internet, the number of simultaneous connections between two IPN nodes will not be high in practical IPN Internet scenarios. Hence, it is likely that very few such reports would be required during connection periods.

Note also that the transmitted time information in ATI pairs is only a time duration rather than a time point which would require time synchronization between the nodes.

Furthermore, the calculation of the data rate validity durations τ_j in (73) and (74) also explicitly considers the RTT between the IPN nodes. Therefore, the feedback received from the IRS_r in ATI pairs can be safely used to accurately set the actual transmission rate and hence LFC algorithm is independent of delayed feedback problem.

On the other hand, while the LFC mechanism described above assumes the equal bandwidth share among simultaneous bundle transports, it can also work with other service disciplines such as first-come first-serve (FCFS) or application specific quality-of-service (QoS) priority-based disciplines. In fact, the class of service field in the bundle header [95] can be used for this purpose. One conceivable scenario is that if the expiration time field of a bundle is empty, i.e., $\delta_k = 0$, then the bundle transport request can be treated as a best-effort service in terms of link resources share, while the other requests can be prioritized based on their time constraints. Note that any such service policy change can be incorporated by the LFC mechanism with only a change in the calculation of the data rate, i.e., R_k^j in (75).

6.4.2 Local Packet-Level Reliability

The deep-space channel of IPN Internet links is safely modeled as the memoryless Additive White Gaussian Noise (AWGN) channel [40]. The raw bit-error rate (BER), i.e., bit error rate before channel coding, is experienced to be usually around 10^{-1} in the past deep space missions [12]. However, the space links need to operate at reasonably low error conditions in order to deliver useful scientific data obtained from high cost deep space missions. For this reason, all NASA missions, including the Earth orbit and deep space, design their radio-frequency (RF) systems to provide 10^{-5} or better BER after physical channel coding is applied [93].

In fact, the CCSDS standards for physical layer channel coding and implementation details use the AWGN model for deep space channel [39]. Due to antenna power/gain constraints, drastic signal degradation due to extreme interplanetary distances, and multipath fading; powerful channel coding is mandatory over deep space channel. Hence, the CCSDS defines and recommends possible channel coding techniques suitable for AWGN deep space

channel including convolutional codes and Turbo codes with varying code rates in Telemetry Channel Coding recommendation [39]. After such powerful physical channel coding is applied to deep space channel, most deep space links operate at 10^{-8} BER as also observed in past deep space missions such as WIND and POLAR missions [93].

Conventional wireless networks may usually require both link layer packet-level forward error correction (FEC) and automatic repeat request (ARQ) based reliability mechanisms and a transport layer end-to-end ARQ-based reliability mechanism. However, as discussed in Section 6.3, in the hop-by-hop custodial transport approach, a transport connection for a certain bundle takes place on a single IPN link between two IPN hops at any given time. Furthermore, end-to-end reliability of bundle transport from the originating source IPN node to the final destination is decoupled from the transmission reliability of a bundle between two IPN hops and is addressed by the bundling layer mechanisms [33]. Therefore, considering powerful channel coding schemes used at the physical layer [42] and the nature of the hop-by-hop transport approach; instead of deploying separate transport and link layer reliability mechanisms on top of a channel coding, a unified packet-level reliability mechanism is adequate in addressing the local packet-level transmission reliability requirement.

Therefore, ITP protocol incorporates a unified local packet-level ARQ-based reliability mechanism into protocol operation as shown in Figure 87. It adopts selective-acknowledgments (SACK) [81] to address burst packet losses which are shown to be beneficial and effective in recovering from multiple packet losses in one round-trip time (RTT) especially in links with high bandwidth-delay products such as deep space links. Furthermore, due to possible inadequacy of the number of SACK blocks in the SACK option field for very long blackout durations, timeout mechanism is also included in ITP.

6.4.3 Blackout Mitigation Procedure

The hop-by-hop bundle transport approach is mainly adopted by the ITP to effectively address the intermittent connectivity challenge imposed by the IPN Internet. However, it is still possible to experience link outages due to loss of line-of-sight by orbital obscurations

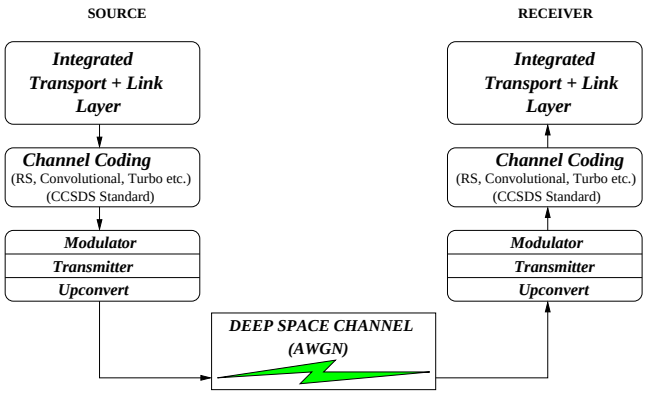

Figure 87: An illustration of ITP deployment for unified packet-level reliability on top of physical layer with channel coding over deep space channel.

which lead to burst packet losses and decrease in the throughput. To reduce the degradation in link utilization due to blackouts, *Blackout Mitigation (BM) Procedure* is developed and incorporated into the ITP protocol operation.

ITP sender receives data ACK packets used mainly for reliability control purposes and also as means of explicit rate control feedback from the receiver carrying the ATI pairs as described in Section 6.4.1. If it does not receive any ACK for a certain period of time T_w, it infers this condition as blackout and invokes BM procedure. Similarly, the same blackout event is also detected by the ITP receiver if it does not receive any data packet for a duration of T_w. The objective of the BM procedure is to reduce the throughput degradation due to the blackout situation.

During BM procedure, ITP source ceases sending data packets. This is mainly because power efficiency is critical in deep space environments. Furthermore, continuously transmitting data packets during a blackout period of unknown length may lead to significant performance degradation due to high number of retransmissions required for reliability purposes after the blackout is over. As the ITP receiver does not receive any data packet for a duration of T_w, it also invokes BM procedure at its end. Although it does not receive any data packets to send ACK for, it keeps sending ACKs with their *BM bit* set to indicate that these ACKs are transmitted as part of blackout mitigation procedure, and hence they are called *BM ACKs*. The objective of BM ACKs is to help ITP source to capture accurate information regarding the blackout situation and act accordingly.

Since RTT is very high, the effect of blackout on the performance changes with the relative location of blackout occurrence in time with respect to the receiver. Let $t = t_0$ be the time when blackout occurs and L is the duration of the blackout. Assume that the blackout occurs at a position x seconds away from the ITP receiver. For $rtt = RTT/2$, there are two distinct cases observed at the ITP sender side according to the duration of the blackout and its relative distance to the ITP receiver in time:

1. $L < 2x$: After $rtt - x$ from t_0, i.e., at $t_1 = t_0 + rtt - x$, ITP sender detects the period with no ACKs. If the duration of the period with no ACK takes more than T_w, then the ITP source invokes BM procedure at $t = t_1$, as in Figure 88.

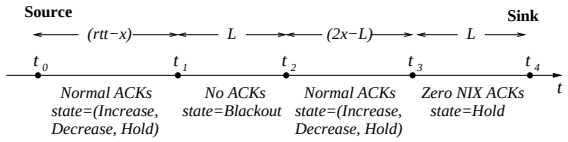

Figure 88: Blackout condition observed from ITP source for $L < 2x$.

In this case, ITP source does not send any new data packets and does not perform any data rate update. At $t_2 = t_1 + L$, ITP source receives normal data ACKs for a duration of $2x - L$. Therefore, ITP source infers that blackout is over, starts retransmission of the packets whose retransmission timer is expired, and resumes its normal LFC algorithm as described in Section 6.4.1. At $t_3 = t_2 + 2x - L$, the source starts to receive BM ACKs for duration of L. These BM ACKs, are in fact, transmitted by the receiver when it detects the same blackout condition. Therefore, ITP sender does not invoke another BM procedure, which would be done otherwise if BM ACKs were not incorporated into BM procedure at the ITP receiver end. Consequently, ITP reduces the effect of blackout on the performance by not wasting the link resources for a duration of L by the help of BM procedure.

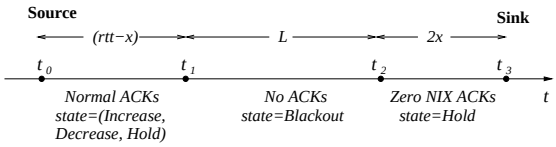

Figure 89: Blackout condition observed from TP-Planet source for $L \geq 2x$.

2. $L \geq 2x$: In this case, ITP source detects no ACK period and invokes BM procedure at $t_1 = t_0 + rtt - x$ for the blackout occurred at $t = t_0$. It again does not perform any rate update and does not send any new data packet. At $t_2 = t_1 + L$, the source starts to receive BM ACKs for duration $2x$ indicating that the blackout is over for ITP source as shown in Figure 89. After that, ITP source starts retransmitting lost packets and resumes the normal LFC algorithm operation. Hence, the duration of $2x$ is efficiently utilized by the help of BM procedure.

Consequently, by incorporating the BM procedure, ITP reduces the throughput degradation due to blackout conditions and improves the link utilization for duration of L or $2x$ in the cases $L < 2x$ and $L \geq 2x$, respectively.

6.4.4 Optimum Packet Size

As previously shown in [2], extreme propagation delay decreases the link utilization in the IPN Internet links. It is also well known that transmission efficiency increases with packet size. On the other hand, an increase in the packet size also leads to an increase in the experienced packet loss probability for a given bit-error rate. Furthermore, packet size also affects the transmission overhead due to reliability mechanism, i.e., the amount of bandwidth consumed in error control by ARQ retransmissions of lost packets for 100% reliability assurance.

Consequently, there exists a tradeoff between packet size, packet loss probability, and transmission efficiency. Hence, the *optimum packet size* is determined in order to achieve maximal compensation for the experienced degradation in link transmission efficiency due to extreme propagation delay subject to packet loss probability and transmission overhead due to ARQ error control mechanism over deep space channel modeled as the AWGN channel as discussed in Section 6.4.2. A closed form solution for such packet size is analytically obtained and then used for the integrated transport and link layer transmission operation of ITP protocol.

Let l be the packet size and θ be the bit-error-rate. The packet loss probability associated

with this packet size, i.e., $p(l)$, is then calculated by

$$p(l) = 1 - (1 - \theta)^l \tag{76}$$

Note that p increases with increasing l as discussed above and hence the objective is to minimize $p(l)$.

On the other hand, the packet size l also affects the transmission efficiency. Let S be the size of a bundle which can be packetized into D packets, i.e., $D = S/l$. The objective of high transmission efficiency is then expressed by

$$\frac{E[\mathcal{N}_{hbh}]}{D} \approx 1 \tag{77}$$

where $E[\mathcal{N}_{hbh}]$ is the expected total number of packets transmitted to reliably transport a bundle of size S including retransmissions as in (69). Hence, the objective is to minimize the reliability transmission overhead, $O_r(l)$, i.e.,

$$O_r(l) = \sum_{j=1}^{N} \sum_{i=0}^{\infty} (1 - p_j(l))\, p_j^i(l)(i+1) - 1 \tag{78}$$

In addition, let $T(l)$ be the transmission efficiency of ARQ mechanism over AWGN deep space channel which is to be maximized. Let h be the packet header, b be the binary transmission rate of AWGN channel, then the $T(l)$ can be expressed by

$$T(l) = \frac{l - h}{l} b \left[f(E_b/N_0) \right]^l \tag{79}$$

where $f(\gamma)$ is the probability of receiving a packet correctly as a function of signal-to-noise ratio (SNR) γ, i.e., E_b/N_0 where E_b is the received energy per bit and N_0 is the receiver noise power spectral density of the AWGN channel.

Therefore, the overall objective is to determine l such that $p(l)$ and $O_r(l)$ are minimized and $T(l)$ is maximized. Hence, it follows from (76), (78), and (79) that the optimum packet size achieving these objectives can be calculated by

$$l = \frac{1}{2} \left[h + \left(h^2 - \frac{4h}{\ln(1 - \theta)} \right)^{1/2} \right] \tag{80}$$

Consequently, (80) can be used to determine the optimum packet size to be used in order to compensate throughput degradation in the IPN links. As discussed in Section 6.4.2, the

deep space missions operate generally at 10^{-8} BER after powerful physical layer channel coding such as Turbo codes is applied. Hence, for $\theta = 10^{-8}$, and the header size of $h = 20$ Bytes, the optimum packet size l is obtained to be approximately 15KB.

6.4.5 Bandwidth Asymmetry

As discussed in Section 6.4.2, ITP uses selective acknowledgments (SACK) [81] for assurance of reliable data segment transmission. Considering the packet size of 3KB determined in Section 6.4.4 and size of SACK packets as 40B, then the ratio of the traffic in the forward and reverse channels is 75:1, i.e., 3KB/40B. Thus, the bandwidth asymmetry up to 75:1 cause no congestion in the reverse link if the ITP receiver continuously sends a SACK for a packet it receives. However, the bandwidth asymmetry in deep space links is usually on the order of 1000:1 [44]. Thus, sending one SACK for each data packet can cause reverse channel to be congested causing SACK losses and hence performance degradation.

To overcome the bandwidth asymmetry problem, ITP regulates SACK traffic on the reverse link by delaying the SACKs. For this purpose, ITP receiver maintains delayed-SACK factor, d, and sends one SACK for every d data packets received. If there is no packet loss and hence no change in the SACK blocks, ITP receiver delays SACKs with a delayed-SACK factor of d. Otherwise, it immediately sends a new SACK with an updated block. Therefore, the amount of traffic on the reverse channel is controlled by adjusting the delayed-SACK factor d. Effects of the bandwidth asymmetry on the throughput performance and the improvement achieved by delayed-SACK is evaluated in Section 6.5.4.

6.5 Performance Evaluation

In order to investigate the performance of the ITP, extensive simulation experiments are conducted. The throughput performance of ITP is analyzed along with the transmission latency experiments in Section 6.5.1. The effects of blackout conditions on the performance and the performance of ITP with the improvement by Blackout Mitigation procedure are investigated in Section 6.5.2. The optimum packet size analysis and its effects on the throughput performance is observed in Section 6.5.3. ITP performance on deep space links with asymmetrical bandwidth and the improvement with delayed SACK are explored in

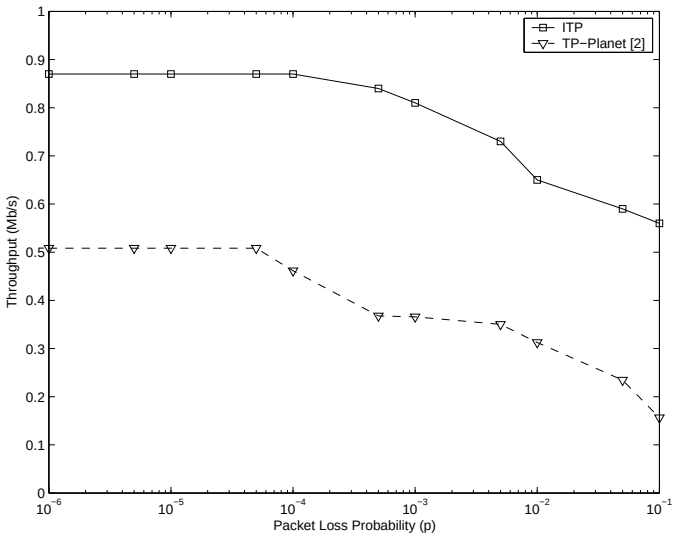

Figure 90: Throughput performance for varying p and $RTT = 600$ seconds, $N = 3$, $B = 100$MB.

Section 6.5.4.

6.5.1 Throughput Performance

In order to show the throughput performance of ITP in the IPN Internet, several simulation experiments are performed by varying packet loss probability p, number of IPN Internet hops N along the end-to-end path, total size of the data bundle to be transmitted B, and the round-trip time RTT. It is assumed that the capacity of the link is 1Mb/s and $RTT = 600$ seconds unless otherwise stated considering the simulation topology shown in Figure 76. Since TP-Planet [6] is the only reliable data transport protocol proposed for the IPN Internet and has been shown to improve the throughput performance with several orders of magnitude over the existing conventional transport protocols [6], ITP performance is also compared with the TP-Planet [6].

The first set of ITP throughput performance experiments are performed for varying packet loss probability p between 10^{-6} and 10^{-1}. As shown in Figure 90, the throughput achieved by both ITP and TP-Planet [6] decrease with increasing packet loss probability p. However, the throughput degradation experienced by the ITP is not as severe as it is observed in case of existing TCP protocols [2] and TP-Planet [6] as shown in Figure 90. This is achieved because of the ITP's explicit rate-based LFC algorithm which is decoupled from

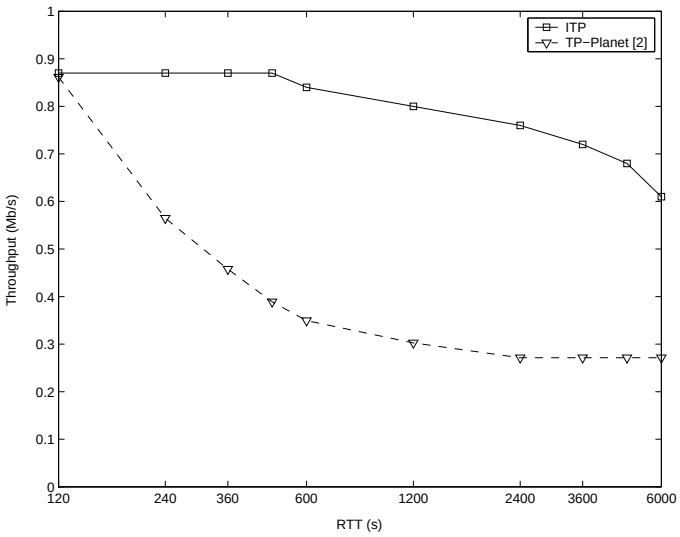

Figure 91: Throughput performance for varying RTT and $p = 10^{-4}$, $N = 3$, $B = 100$MB.

the packet loss and reliability mechanism as described in Section 6.4.1. For transmission of 100MB and $p = 10^{-4}$, ITP achieves throughput of 870Kb/s while TP-Planet can only achieve 461 Kb/s. Hence, ITP significantly outperforms the TP-Planet [6] for a wide range of packet loss probability p.

In Figure 91, the effect of RTT on the throughput performance is shown. The simulation experiments are performed for the transmission of a 50MB data bundle, $p = 10^{-4}$, $N = 3$ and varying RTT between 120 to 6000 seconds. An increase in RTT leads to a slight degradation in the throughput performance of ITP as shown in Figure 91. However, TP-Planet experiences higher throughput loss with increasing end-to-end path delay. This is mainly because the ITP deploys the new local flow control algorithm based on explicit rate allocation rather than the additive-increase multiplicative-decrease mechanisms whose throughput performance is directly affected by the RTT as observed with the TP-Planet performance. For $RTT = 600$ seconds, ITP achieves approximately 800 Kb/s throughput which corresponds to 140% performance improvement over TP-Planet. Note that this throughput value further increases with increasing bundle size B as observed in Figure 92.

ITP simulation experiments are also performed for varying data bundle size B between 1 and 1000 MB. As shown in Figure 92, the throughput increases with increasing data bundle size for both ITP and TP-Planet. This is because the connection period increases

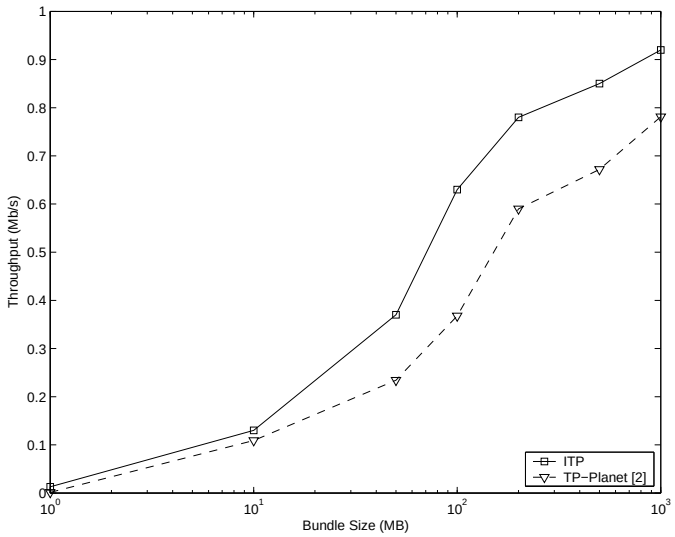

Figure 92: Throughput performance for varying B and $RTT = 600$ seconds, $p = 10^{-4}$, $N = 3$.

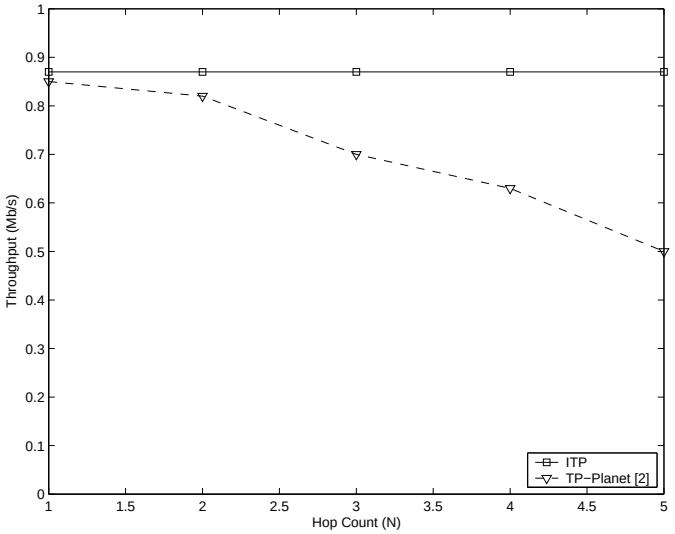

Figure 93: Throughput performance for varying N and $RTT = 600$ seconds, $p = 10^{-4}$, $B = 100$MB.

with the bundle size which, in turn, decreases the effects of end-to-end path delay over the throughput performance. For a bundle size of 1000MB, ITP achieves 920 Kb/s throughput out of 1 Mb/s link capacity.

ITP simulation experiments are also performed for varying number of IPN hops N involving in the end-to-end connection path. Here, N is varied between 1 and 5. As observed in Figure 93, the throughput performance of ITP is not affected by the hop count. Note that

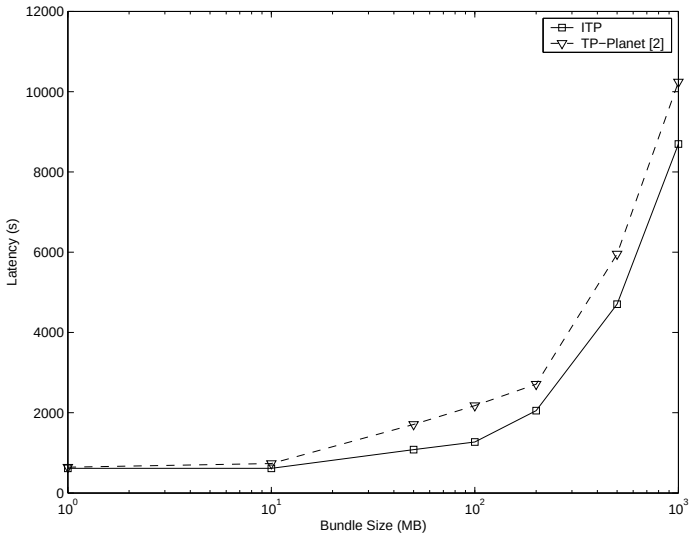

Figure 94: Transmission latency for varying B, $p = 10^{-4}$, $N = 3$.

this is achieved because of the hop-by-hop nature of the ITP operation. However, TP-Planet performance decreases as the number of hops involving the end-to-end path increases. This is mainly because TP-Planet performs end-to-end reliability and congestion control which are respectively subject to inefficiency and inaccuracy as the IPN path increases. Note that IPN paths involving in several IPN hops may correspond to communication paths between the Earth and the farther planets such as Jupiter and Pluto. For $N = 5$, ITP outperforms TP-Planet with 74% throughput improvement. Note also that these results are also consistent with the initial analytical study for the transmission effiency comparison of the end-to-end and hop-by-hop approaches as in Section 6.3.

In 94, the time it takes for ITP and TP-Planet to successfully transport a bundle of B MB from the source to the final destination is also shown, i.e., transmission latency. As shown in Figure 94, the latency experienced increases with bundle size as expected. However, note that transmission latency experienced by ITP is lower than TP-Planet. This is because TP-Planet link utilization is affected by longer retransmission paths and end-to-end delays as also observed in Figure 92. For a bundle size of 1000MB, ITP has approximately 18% lower transmission latency than TP-Planet.

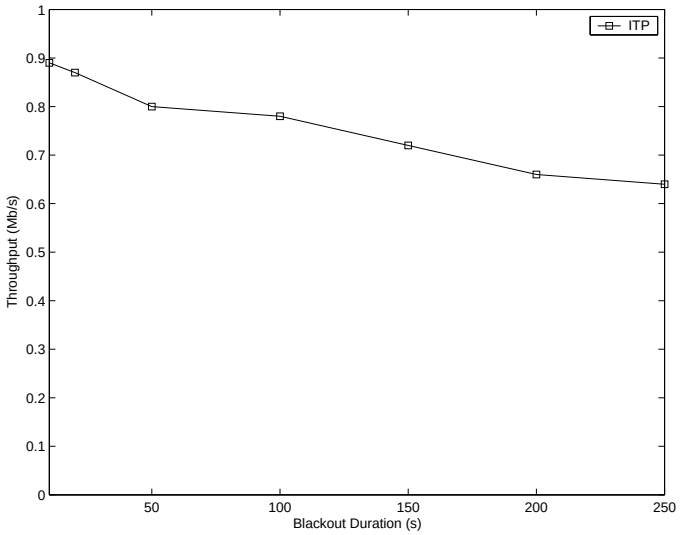

Figure 95: Throughput performance for varying blackout duration.

6.5.2 Blackout Performance

When a blackout is detected, ITP invokes Blackout Mitigation (BM) procedure in order to reduce its effect on the throughput performance as explained in Section 6.4.3. Throughput achieved by ITP for different blackout durations is given in Figure 95. Here, $RTT = 120$ seconds, $p = 10^{-5}$. The simulations are performed for a duration of 600 seconds, where the blackout occurs at $t = 250$ seconds.

As in Figure 95, throughput decreases with increasing blackout duration as expected. However, BM procedure improves the performance for long blackout durations. For even a blackout of 150 seconds, which is 1/4 of the entire simulation time, ITP achieves 720 Kb/s throughput over the space link with 1Mb/s with the help of the BM procedure.

6.5.3 Packet Size

In Section 6.4.4, the optimum packet size for ITP is analytically determined in order to compensate for the throughput degradation experienced in the IPN Internet links due to high propagation delay and packet losses. In order to observe the optimum packet size and its effect on the throughput performance, (80) is plot and simulation experiments are also performed for varying packet size.

As observed in Figure 96, the optimum packet size l in (80) decreases with increasing

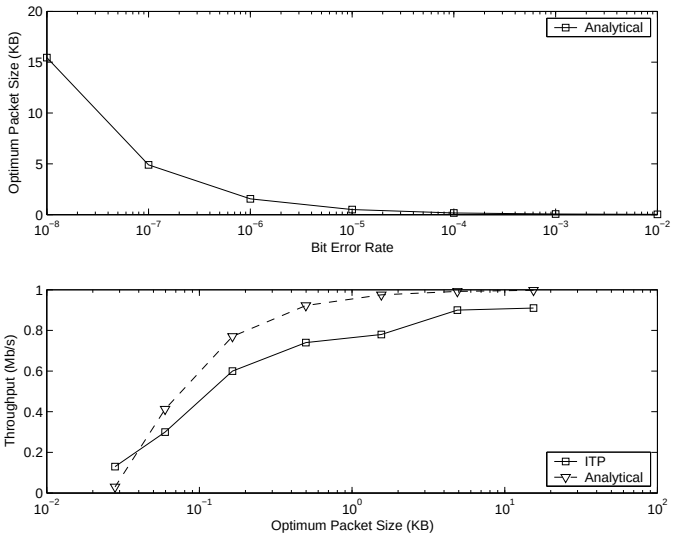

Figure 96: Optimum packet size and its effect on the ITP throughput performance.

bit error rate, i.e., θ in (80). This is consistent with the objective of minimizing ARQ retransmission overhead by minimizing packet loss probability which was considered as one of the constraints in the determination of the optimum packet size in Section 6.4.4. For the obtained optimum packet size values, the throughput efficiency in (79) is analytically observed and also simulation experiments are run using the same topology and parameters described in Section 6.5.1. As shown in Figure 96, throughput increases with the packet size in both analytical and simulation cases. However, the increase in the throughput decelerates as the packet size becomes larger. This is because the packet loss probability for a given bit error rate also increases leading to an increase in the number of retransmissions and hence a decrease in the throughput performance. Consequently, the optimum packet size determined in Section 6.4.4 helps ITP to maximize its throughput efficiency in the IPN Internet links.

6.5.4 Bandwidth Asymmetry Performance

Here, simulation experiments are performed to show the effect of bandwidth asymmetry on the performance and the improvement achieved by delayed SACK. $RTT = 120$ seconds $p = 10^{-4}$ and the simulation time is assumed to be 1200 seconds. In Figure 97, two cases with different bandwidth asymmetry ratio are investigated: forward and reverse link

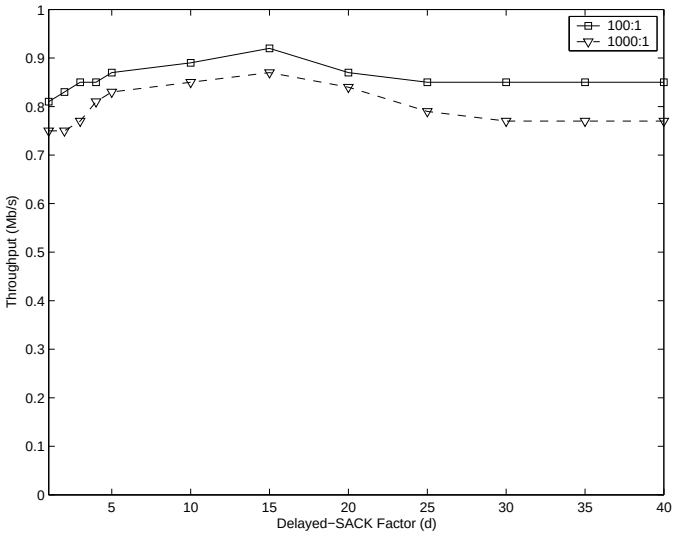

Figure 97: Throughput performance for varying bandwidth asymmetry ratio.

capacities are 1000 Kb/s and 10 Kb/s, i.e., 100:1; and 1000 Kb/s 1Kb/s, 1000:1, i.e., respectively.

As observed from both of the curves with different bandwidth asymmetry levels, i.e., 100:1 and 1000:1, in Figure 97, the throughput is not significantly affected by asymmetrical channel capacity of the deep space link. This is mainly because unlike the conventional ACK-controlled congestion control mechanisms, the flow control operation maintained by LFC algorithm is not regulated by the ACKs. Therefore, congestion in the reverse channel does not directly hit the throughput performance.

As shown in Figure 97, an increase in the delayed SACK factor also leads to a slight increase in the throughput. However, throughput decreases for both cases with either too high or too small delayed SACK factor of d. This is because if it is too small, reverse channel is congested. Hence, loss of the ACKs carrying the ATI pairs may lead to inaccuracy in the LFC operation and therefore this may result in some decrease in link utilization. On the other hand, if d is too large, the source receives less number of SACKs carrying ATI pairs than it expects which leads to performance degradation. Thus, at $d = 15$, throughput achieved increases up to 900Kb/s.

CHAPTER VII

EVENT-TO-SINK RELIABLE TRANSPORT IN WIRELESS SENSOR NETWORKS

In this chapter, a new reliable transport scheme for wireless sensor networks (WSN), the event-to-sink reliable transport (ESRT) protocol, is presented. ESRT is a novel transport solution developed to achieve reliable event detection in WSN with minimum energy expenditure. The ESRT protocol was first presented in [92] and then revised and further enhanced in [5] to accommodate the scenarios where multiple concurrent events occur in the wireless sensor field. In Section 7.2, a review of related work in transport protocols, both in WSN and other communication networks, are presented along with their inadequacies. The transport problem in WSN is formally defined in Section 7.3 and five characteristic reliability regions are identified. These regions determine the appropriate actions taken by ESRT. The operation of ESRT is described in detail in Section 7.4 and a pseudo-algorithm is also presented. In Section 7.5, how the default ESRT protocol operation is extended to accommodate the scenarios where multiple concurrent events occur in the wireless sensor field is described. ESRT performance analysis and simulation results are presented in Section 7.6.

7.1 Motivation

Wireless Sensor Network (WSN) is an event driven paradigm that relies on the collective effort of numerous microsensor nodes. This has several advantages over traditional sensing including greater accuracy, larger coverage area and extraction of localized features. In order to realize these potential gains, it is imperative that desired event features are reliably communicated to the sink.

To accomplish this, a reliable transport mechanism is required in addition to robust

modulation and media access, link error control and fault tolerant routing. The functionalities and design of a suitable transport solution for WSN are the main issues addressed in this work.

The need for a transport layer for data delivery in WSN was questioned in a recent work [118] under the premise that data flows from source to sink are generally loss tolerant. While the need for end-to-end reliability may not exist due to the sheer amount of correlated data flows, an event in the sensor field needs to be tracked with a certain accuracy at the sink. Hence, unlike traditional communication networks, the sensor network paradigm necessitates an *event-to-sink* reliability notion at the transport layer. This is a truly novel aspect of this work and is the main theme of the proposed Event-To-Sink Reliable Transport (ESRT) protocol for WSN. Such a notion of collective identification of data flows from the event to the sink is illustrated in Figure 98.

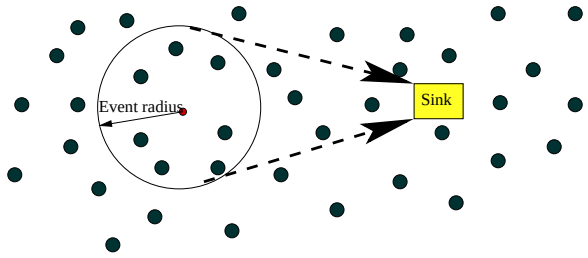

Figure 98: Typical sensor network topology with event and sink. The sink is only interested in collective information of sensor nodes within the event radius and not in their individual data.

This work is also motivated by the results in [112], which emphasize the need for congestion control in WSN. It was shown in [112] that exceeding network capacity can be detrimental to the observed goodput. However, the authors stopped short of providing a solution to this problem.

ESRT is a novel transport solution that seeks to *achieve reliable event detection with minimum energy expenditure and congestion resolution.* It has been tailored to match the unique requirements of WSN. Some of its salient features are

1. *Self-configuration* - Reliable event detection must be established and maintained in the face of dynamic topology in WSN. Topology dynamics can result from either the

failure or temporary power-down of energy constrained sensor nodes. Spatial variation of events and random node deployment only exacerbate the above problem. ESRT is self-configuring and achieves flexibility under dynamic topologies by self-adjusting the operating point (see Section 7.4).

2. *Energy awareness* - Although the primary goal of ESRT is reliable event detection, it aims to accomplish this with minimum possible energy expenditure. For instance, if reliability levels at the sink are found to be in excess of that required, the source nodes can conserve energy by reducing their reporting rate (see Section 7.4).

3. *Congestion Control* - Packet loss due to congestion can impair event detection at the sink even when enough information is sent out by the sources. Hence, congestion control is an important component for reliable event detection in WSN. An important feature of ESRT is that congestion control is also used to reduce energy consumption. Correlated data flows are loss tolerant to the extent that event features are reliably communicated to the sink. Due to this unique characteristic of WSN, required event detection accuracy may be attained even in the presence of packet loss due to network congestion. In such cases however, a suitable congestion control mechanism can help conserve energy while maintaining desired accuracy levels at the sink. This is done by conservatively reducing the reporting rate. Details of such a mechanism are presented in Section 7.4.

4. *Collective identification* - In typical WSN applications, the sink is only interested in the collective information provided by numerous sensor nodes and not in their individual reports. In accordance with this, ESRT does not require individual node IDs for operation. This is also in tune with the proposed event-to-sink model rather than the traditional end-to-end model. More importantly, this can ease implementation costs and reduce overhead.

5. *Biased Implementation* - The algorithms of ESRT mainly run on the sink with minimum functionalities required at sensor nodes. This helps conserve limited sensor

resources and shifts the burden to the high-powered sink. Such a graceful transfer of complexity is possible only due to the event-to-sink reliability notion.

ESRT is designed for use in typical WSN applications involving event detection and signal estimation/tracking, and not for guaranteed end-to-end data delivery services. This work is motivated by the fact that the sink is only interested in reliable detection of event features from the collective information provided by numerous sensor nodes and not in their individual reports. This notion of event-to-sink reliability distinguishes ESRT from other existing transport layer models that focus on end-to-end reliability. The reliable transport in WSN has not been studied from this perspective before.

Furthermore, in this chapter, the work in [92] is also extended by enhancing ESRT protocol in order to accommodate the scenarios where multiple concurrent events occur in the wireless sensor field. Such enhancement is significant since the data flows generated by the multiple events occurring simultaneously may not be always isolated in the wireless sensor network. Thus, uncoordinated protocol actions may fail to achieve required event-to-sink transport reliability and to resolve congestion for individual event flows because of the interaction between these flows in the network. Therefore, it is necessary to accurately capture the event occurrence situation in the network and accordingly act to assure the event-to-sink reliability with minimum energy expenditure for all of the multiple concurrent events in the sensor field.

7.2 Related Work

Despite the considerable amount of research on several aspects of sensor networking, the problems of reliable transport and congestion control are yet to be efficiently studied and addressed. The urgent need for congestion control is pointed out within the discussion of infrastructure tradeoffs for WSN in [112]. However, the authors do not propose any solution for the problem they identify.

In another recent work [118], the PSFQ (Pump Slowly, Fetch Quickly) mechanism is proposed for reliable retasking/ reprogramming in WSN. PSFQ is based on slowly injecting packets into the network, but performing aggressive hop-by-hop recovery in case of packet

loss. The pump operation in PSFQ simply performs controlled flooding and requires each intermediate node to create and maintain a data cache to be used for local loss recovery and in-sequence data delivery. Although this is an important transport layer solution for WSN, it is applicable only for strict sensor-to-sensor reliability and for purposes of control and management in the reverse direction from the sink to sensor nodes. Event detection/tracking in the forward direction does not require guaranteed end-to-end data delivery as in PSFQ. Individual data flows are correlated and loss tolerant to the extent that desired event features are collectively and reliably informed to the sink. Hence, the use of PSFQ for the forward direction can lead to a waste of valuable resources. In addition to this, PSFQ does not address packet loss due to congestion.

In [103], the RMST (Reliable Multi-Segment Transport) protocol is proposed to address the requirements of reliable data transport in wireless sensor networks. RMST is mainly based on the functionalities provided by *directed diffusion* [61]. Furthermore, RMST utilizes in-network caching and provides guaranteed delivery of the data packets generated by the event flows. However, as discussed above, event detection/tracking does not require guaranteed end-to-end data delivery since the individual data flows are correlated loss tolerant. Moreover, such guaranteed reliability via in-network caching may bring significant overhead for the sensor networks with power and processing limitations.

In contrast, ESRT is based on an event-to-sink reliability model and provides reliable event detection without any intermediate caching requirements. ESRT also seeks to achieve the required event detection accuracy using minimum energy expenditure and has a congestion control component.

A novel transmission control scheme for use at the MAC layer in WSN is proposed in [122] with the main objective of per-node fair bandwidth share. Energy efficiency is maintained by controlling the rate at which MAC layer injects packets into the channel. Although such an approach can control the transmission rate of a sensor node, it neither considers congestion control nor addresses reliable event detection. For similar reasons, the use of other MAC protocols like the IEEE 802.11 DCF or S-MAC [128] that provide some form of hop reliability is inadequate for reliable event detection in WSN.

Next, transport solutions in other wireless networks are briefly examined and their inadequacies when applied to WSN are pointed out. These studies mainly focus on reliable data transport following end-to-end TCP semantics and are proposed to address the challenges posed by wireless link errors and mobility [17]. The primary reason for their inapplicability in WSN is their notion of end-to-end reliability. Furthermore, all these protocols bring considerable memory requirements to buffer transmitted packets until they are ACKed by the receiver. In contrast, sensor nodes have limited buffering space (<4KB in MICA motes [82]) and processing capabilities. Hence, there is a need for a novel transport mechanism in WSN that emphasizes on collective reliability, resource efficiency and simplicity.

The multi-hop and many-to-one nature of data flows in WSN prompts a review of reliable multicast solutions proposed in other wired/wireless networks. There exist many such schemes that address the reliable transport and congestion control for the case of single sender and multiple receivers [53]. Although the communication structure of the reverse path, i.e., from sink to sources in WSN, is an example of multicast, it is not valid for the forward channel where multiple correlated reports are sent to a single destination. Similar transport problems with multiple senders and a single receiver in other wired/wireless networks simply corresponds to a multiple unicast. However, the WSN paradigm requires the notion of collective reliability. Hence, neither the reliable multicast nor unicast transport solutions can be applied in this case.

7.3 The Reliable Transport Problem in WSN

In preceding discussions, the notion of event-to-sink reliability in WSN is introduced and the inapplicability of existing transport solutions is pointed out. Before proceeding to discuss the proposed Event-To-Sink Reliable Transport (ESRT) protocol, the reliable transport problem in WSN is formally defined in this section. The evaluation environment used in the studies is also introduced and the stage is set for ESRT by defining five characteristic reliability regions.

7.3.1 Problem Definition

Consider typical WSN applications involving the reliable detection and/or estimation of event features based on the collective reports of several sensor nodes observing the event. Let us assume that for reliable temporal tracking, the sink must decide on the event features every τ time units. Here, τ represents the duration of a decision interval and is fixed by the application. At the end of each decision interval, the sink makes an informed decision based on reports received from sensor nodes during that interval. The specifics of such a decision making process are application dependent and beyond the present scope.

The least that can be assumed is that the sink derives a reliability indicator r_i at the end of decision interval i. Note that r_i must be calculated only using parameters available at the sink. Hence, notions of throughput/goodput (as in [112]), which are based on the number of source packets sent out are inappropriate in this case.

The reliable transport of event features from source nodes to the sink is measured in terms of the number of received data packets. Regardless of any application-specific metric that may actually be used, the number of received data packets is closely related to the amount of information acquired by the sink for the detection and extraction of event features. Hence, this serves as a simple but adequate reliability measure at the transport level. The observed and desired event reliabilities are now defined as follows :

Definition 1 *The* observed event reliability*, r_i, is the number of received data packets in decision interval i at the sink*

Definition 2 *The* desired event reliability*, R, is the number of data packets required for reliable event detection. This is determined by the application*

If the observed event reliability, r_i, is greater than the desired reliability, R, then the event is deemed to be reliably detected. Else, appropriate action needs to be taken to achieve the desired reliability, R.

With the above definition, r_i can be computed by stamping source data packets with an event ID and incrementing the received packet count at the sink each time the ID is detected

in decision interval i[1]. Note that this does not require individual identification of sensor nodes. Further, any increase in source information about the event features is modeled as a corresponding increase in the reporting rate, f, of sensor nodes. The reporting rate of a sensor node is defined as the number of packets sent out per unit time by that node. The transport problem in WSN is to *configure the reporting rate, f, of source nodes so as to achieve the required event detection reliability, R, at the sink with minimum resource utilization.*

7.3.2 Evaluation Environment

In order to study the relationship between the observed reliability at the sink, r, and the reporting frequency, f, of sensor nodes, an evaluation environment is developed using *ns*-2 [111]. The parameters used in this study are listed in Table 1.

200 sensor nodes were randomly positioned in a 100x100 sensor field. Node parameters such as radio range and IFQ (buffer) length were carefully chosen to mirror typical sensor mote values [82]. One of these nodes was chosen as the sink to which all source data was sent. Event centers (X_{ev}, Y_{ev}) were randomly chosen and all sensor nodes within the event radius behave as sources for that event. In order to communicate source data to the sink, a simple CSMA/CA based MAC protocol and Dynamic Source Routing (DSR) [68] are employed. The impact of using other routing protocols on the achieved goodput behavior with reporting period was shown to be insignificant in [112]. Hence, it is reasonable to assume that the r vs. f behavior and ESRT performance are insensitive to the underlying routing protocol.

The results of this study are shown in Figure 99 for number of source nodes $n = 41, 52, 62$. Note that each of these curves was obtained by varying the reporting rate f for a certain event center (X_{ev}, Y_{ev}) and corresponding number of senders n. These values are tabulated in Table 2. The event radius was fixed throughout at $30m$.

The following observations are made from Figure 99

[1]With in-network data aggregation, one must account for data packets that were aggregated en route to the sink

Table 6: NS-2 simulation parameters.

Area of sensor field	100x100 m^2
Number of sensor nodes	200
Radio range of a sensor node	40 m
Packet length	30 bytes
IFQ length	65 packets
Transmit Power	0.660 W
Receive Power	0.395 W
Decision interval (τ)	10 sec

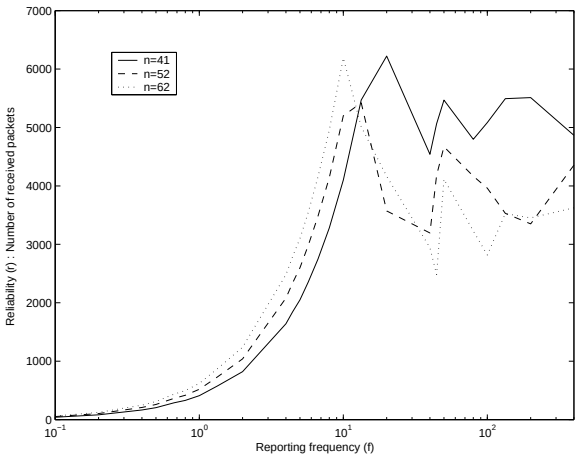

Figure 99: The effect of varying the reporting rate, f, of source nodes on the event reliability, r, observed at the sink. The number of source nodes is denoted by n.

1. The reliability, r, shows a linear increase (note the log scale) with source reporting rate, f, until a certain $f = f_{max}$, beyond which the reliability drops. This is because the network is unable to handle the increased injection of data packets and packets are dropped due to congestion.

2. Such an initial increase and subsequent decrease in reliability is observed regardless of the number of source nodes, n.

3. f_{max} decreases with increasing n, i.e., congestion occurs at lower reporting frequencies with greater number of sources.

4. For $f > f_{max}$, the behavior is rather wavy and not smooth. An intuitive explanation for such a behavior is as follows. The number of received packets, which is the reliability, r, is the difference between the total number of source data packets, s, and the

number of packets dropped by the network, d. While s simply scales linearly with f, the relationship between d and f is non-linear. In some cases, the difference $s - d$ is seen to increase eventhough the network is congested. The important point to note however, is that this wavy behavior always stays well below the maximum reliability at $f = f_{max}$

5. The drop in reliability due to network congestion is more significant with increasing n.

Table 7: Event centers for the three curves with n=41,52,62 in Figure 99.

Number of source nodes	**Event Center** (X_{ev},Y_{ev})
41	(88.2,62.8)
52	(32.6,79.3)
62	(39.2,58.1)

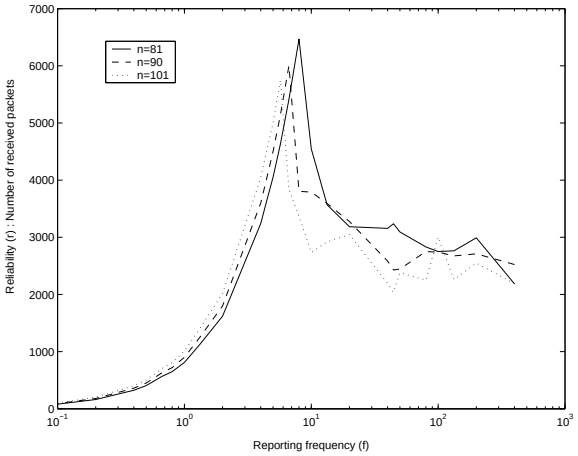

Figure 100: The effect of varying the reporting rate, f, of source nodes on the event reliability, r, observed at the sink. The number of source nodes is denoted by n.

Figure 100 shows a similar trend between r and f with further increase in n ($n = 81, 90, 101$). As before, the event centers are tabulated in Table 3. The event radius was fixed at $40m$ for this set of experiments.

The wavy behavior for $f > f_{max}$ observed in Figure 99 persists in Figure 100, but appears rather subdued because of much steeper drops due to congestion (see observation

Table 8: Event centers for the three curves with n=81,90,101 in Figure 100.

Number of source nodes	Event Center (X_{ev},Y_{ev})
81	(32.6,79.3)
90	(61.1,31.5)
101	(60.0,63.6)

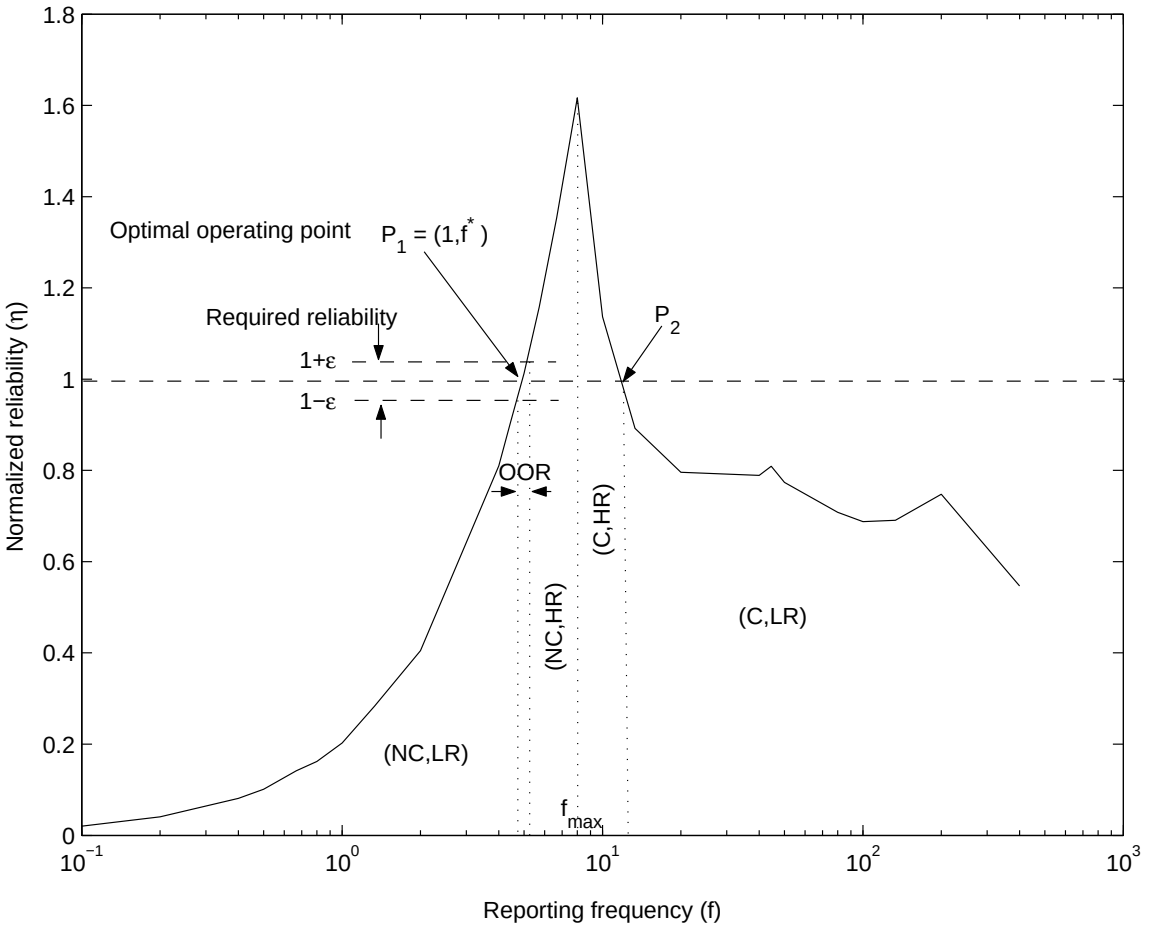

Figure 101: The five characteristic regions in the normalized reliability, η, vs. reporting frequency, f, behavior.

5 earlier). All the other trends observed earlier are confirmed in Figure 100.

7.3.3 Characteristic Regions

A general trend of initial reliability, r, increase with reporting frequency, f, and subsequent decrease due to congestion loss is evident from the preliminary studies in Figure 99 and Figure 100. This confirms the urgent need for an event-to-sink reliable transport solution with

a congestion control mechanism in WSN. The r vs. f characteristics is further elaborated next and five characteristic regions are identified. As will be seen shortly, these regions are important for the operation of ESRT.

Consider a representative curve from Figure 100 for $n = 81$ senders. This is replicated for convenience in Figure 101. All the subsequent discussions use this particular case for illustration. However, it was verified that the r vs. f behavior shows the general trend of initial increase and subsequent decrease due to congestion regardless of the parameter values. This is indeed observed in Figs. 99 and 100 for varying values of n. Hence, the discussions and results in this work apply to a general r vs. f behavior in WSN with any set of parameter values, with the specific case ($n = 81$) used only for illustration purposes.

Let the desired reliability as laid down by the application be R. Hence, a normalized measure of reliability is $\eta = \frac{r}{R}$. As before, η_i denotes the normalized reliability at the end of decision interval i.

The aim is to operate as close to $\eta = 1$ as possible, while utilizing minimum network resources (f close to f^* in Figure 101). This is called the *optimal operating point*, marked as P_1 in Figure 101. For practical purposes, a tolerance zone of width 2ϵ around P_1 is defined as shown in Figure 101. Here, ϵ is a protocol parameter. The suitable choice of ϵ and its impact on ESRT protocol operation is dealt with in Section 7.6.3.

Note that the $\eta = 1$ line intersects the reliability curve at two distinct points P_1 and P_2 in Figure 101. Though the event is reliably detected at P_2, the network is congested and some source data packets are lost. Event reliability is achieved only because the high reporting frequency of source nodes compensates for this congestion loss. However, this is a waste of limited energy reserves and hence is not the optimal operating point. Similar reasoning holds for $\eta > 1 + \epsilon$.

From Figure 101, five characteristic regions (bounded by dotted lines) are identified using the following decision boundaries

- **(NC,LR)** : $f < f_{max}$ and $\eta < 1 - \epsilon$ (No Congestion, Low Reliability)
- **(NC,HR)** : $f \leq f_{max}$ and $\eta > 1 + \epsilon$ (No Congestion, High Reliability)

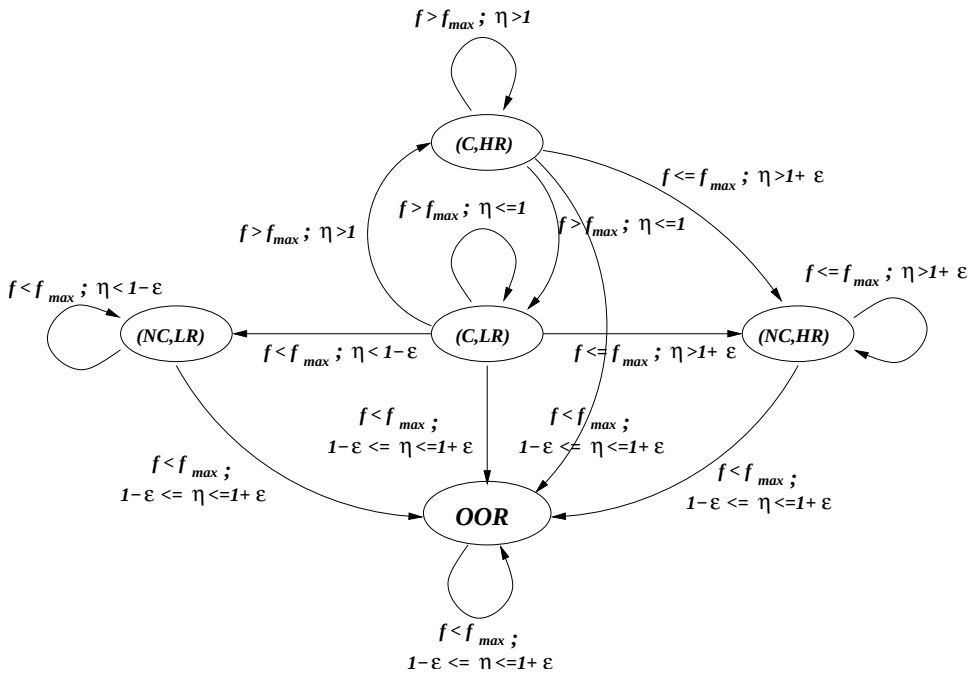

Figure 102: ESRT protocol state model and transitions.

- **(C,HR)** : $f > f_{max}$ and $\eta > 1$ (Congestion, High Reliability)
- **(C,LR)** : $f > f_{max}$ and $\eta \leq 1$ (Congestion, Low Reliability)
- **OOR** : $f < f_{max}$ and $1 - \epsilon \leq \eta \leq 1 + \epsilon$ (Optimal Operating Region)

As seen earlier, the sink derives a reliability indicator η_i at the end of decision interval i. Coupled with a congestion detection mechanism (to determine $f \gtrless f_{max}$), this can help the sink determine in which of the above regions the network currently resides. Hence, these characteristic regions identify the state of the network. Let $\mathbf{S}_i$ denote the network state variable at the end of decision interval i. Then,

$$\mathbf{S}_i \in \{\textbf{(NC,LR),(NC,HR),(C,HR),(C,LR),OOR}\}$$

The operation of ESRT is closely tied to the current network state $\mathbf{S}_i$. The ESRT protocol state model and transitions are shown in Figure 102. Next, the specifics of ESRT and its operation in each of these states are discussed in detail.

7.4 ESRT: Event-To-Sink Reliable Transport Protocol

ESRT is a novel solution that is proposed to address the transport problem in WSN. The primary motive of ESRT is to achieve and maintain operation in state **OOR**. Hence, the aim is to configure the reporting frequency f to achieve the desired event detection accuracy with minimum energy expenditure. To help accomplish this, ESRT uses a congestion control mechanism that serves the dual purpose of reliable detection and energy conservation.

Recall that the r vs. f characteristic shown in Figure 101 can change with dynamic topology resulting from either the failure or temporary power-down of sensor nodes. Hence, an efficient transport protocol should keep track of the reliability observed at the sink and accordingly configure the operating point. If η_i is within the desired reliability limits ($1 - \epsilon \leq \eta_i \leq 1 + \epsilon$) and no congestion notification alert is received, then state **OOR** has been reached and the sink informs source nodes to maintain the current reporting frequency f_i. Here, it is reasonably assumed that the sink is powerful enough to reach all source nodes by broadcast.

In general, the network can reside in any one of the five states $\mathbf{S}_i \in$ **{(NC,LR), (NC,HR), (C,HR), (C,LR), OOR}**. Depending on the current state $\mathbf{S}_i$, ESRT calculates an updated reporting frequency f_{i+1}, which is then broadcast to the source nodes. For example, if $\mathbf{S}_i \in$ **{(NC,LR),(C,LR)}**, the observed reliability levels are inadequate to detect the desired event features. In such a case, ESRT aggressively updates the reporting frequency to reliably track the event as soon as possible.

This self-configuring nature of ESRT helps it adapt to dynamic topology and random deployment, both typical of WSN. Another important feature of ESRT is its inclination to conserve scarce energy resources when reliability levels exceed those required for event detection. This is the case when $\mathbf{S}_i \in$ **{(NC,HR),(C,HR)}**. The motivation to reduce the reporting frequency in this case comes from energy conservation. However, the primary motive of reliable event detection must not be compromised. Hence, ESRT takes a conservative approach in this case and decreases f in a controlled manner.

The algorithms of ESRT mainly run on the sink, with minimal functionality at the source nodes. More precisely, sensor nodes only need the following two additional functionalities

- Sensor nodes must listen to the sink broadcast at the end of each decision interval and update their reporting rates
- Sensor nodes must deploy a simple and overhead-free local congestion detection support mechanism

While the former is an implementation issue and is not within the scope of this work, the details of a congestion detection mechanism are provided in Section 7.4.2. Such a graceful transfer of complexity from sensor nodes to the sink node reduces management costs and saves on valuable sensor resources. Further simplifying implementation is the fact that ESRT works on the collective identification principle and does not require unique source IDs.

In the following subsection, the operation of ESRT in each network state is discussed and a pseudo-algorithm for its implementation is presented.

7.4.1 ESRT Protocol Operation

ESRT identifies the current state $\mathbf{S}_i$ from

- Reliability indicator η_i computed by the sink for decision interval i
- A congestion detection mechanism,

using the decision boundaries defined in Section 7.3.3. Depending on the current state $\mathbf{S}_i$, and the values of f_i and η_i, ESRT then calculates the updated reporting frequency f_{i+1} to be broadcast to the source nodes. At the end of the next decision interval, the sink derives a new reliability indicator η_{i+1} corresponding to the updated reporting frequency f_{i+1} of source nodes. In conjunction with any congestion reports, ESRT then determines the new network state $\mathbf{S}_{i+1}$. This process is repeated until the optimal operating region (state **OOR**) is reached. The state model of the ESRT protocol and state transitions are shown in Figure 102. Note that not all transitions between states are possible, as explained in Section 7.6.1. This is due to the frequency update policies adopted by ESRT, which are now described in detail for each of the five states.

1. **(NC,LR)** *(No Congestion, Low Reliability)* : In this state, no congestion is experienced and the achieved reliability is lower than that required, i.e., $\eta < 1 - \epsilon$ and $f < f_{max}$. This can be the result of one/more of the following

   - Failure/power-down of intermediate routing nodes
   - Packet loss due to link errors
   - Inadequate information sent by source nodes

   When intermediate nodes fail/power-down, packets that need to be routed through these nodes are dropped. This can cause a drop in reliability even if enough source information is sent out. However, fault-tolerant routing/re-routing in WSN is provided by several existing routing algorithms [61, 96]. ESRT can work with any of these routing schemes.

   Packet loss due to link errors may be fairly significant in WSN due to the energy inefficiency of powerful error correction [98] and retransmission techniques. However, regardless of the packet error rate, the total number of packets lost due to link errors is expected to scale proportionally with the reporting frequency f. Here, it is assumed that the net effect of channel conditions on packet loss does not deviate appreciably in successive decision intervals. This is reasonable with static sensor nodes, slowly time-varying ([98, 101]) and spatially separated channels for communication from event-to-sink in WSN applications. Hence, even in the presence of packet loss due to link errors, the initial reliability increase (Observation 1, Section 7.3.2) is expected to be linear.

   It is now clear that in order to improve the reliability to acceptable levels, it is necessary to increase the source information. Since the primary objective of ESRT is to achieve event-to-sink reliability, the reporting frequency f is aggressively increased to attain the required reliability as soon as possible. Such an aggressive increase can be achieved by invoking the fact that the r vs. f relationship in the absence of congestion, i.e., for $f < f_{max}$, is linear. This prompts the use of the following multiplicative

increase strategy to calculate reporting rate update f_{i+1}

$$f_{i+1} = \frac{f_i}{\eta_i} \tag{81}$$

where η_i is the reliability observed at the sink at the end of decision interval i.

2. **(NC,HR)** *(No Congestion, High Reliability)* : In this state, the required reliability level is exceeded, and there is no congestion in the network, i.e., $\eta > 1 + \epsilon$ and $f \leq f_{max}$. This is because source nodes report more frequently than required. The most important consequence of this condition is excessive energy consumption by sensor nodes. Therefore the reporting frequency should be reduced in order to conserve energy. However, this reduction must be performed cautiously so that the event-to-sink reliability is always maintained. Hence, the sink reduces reporting frequency f in a controlled manner with half the slope, as opposed to the aggressive approach in the previous case. Intuitively, here, a balance is being stroke between saving the maximum amount of energy and losing reliable event detection. Thus the updated reporting frequency can be expressed as

$$f_{i+1} = \frac{f_i}{2}\left(1 + \frac{1}{\eta_i}\right) \tag{82}$$

It is shown in Section 7.6 that such an update policy reduces the energy consumption in the network and does not compromise on event reliability.

3. **(C,HR)** *(Congestion, High Reliability)* : In this state, the reliability is higher than required, and congestion is experienced, i.e., $\eta > 1$ and $f > f_{max}$. This is due to the unique feature of WSN where required event detection reliability can be attained even when some of the source data packets are lost. In this case ESRT decreases the reporting frequency in order to avoid congestion and conserve energy in sensor nodes. As before, this decrease should be performed carefully such that the event-to-sink reliability is always maintained. However, the network operating in state **(C,HR)** is farther from the optimal operating point than in state **(NC,HR)**. Therefore, it is necessary to take a more aggressive approach so as to relieve congestion and enter

state **(NC,HR)** as soon as possible. This is achieved by emulating the linear behavior of state **(NC,HR)** with the use of multiplicative decrease as follows

$$f_{i+1} = \frac{f_i}{\eta_i} \tag{83}$$

It can be shown that such a multiplicative decrease achieves all objectives (see Section 7.6).

4. **(C,LR)** *(Congestion, Low Reliability)* : In this state the observed reliability is inadequate and congestion is experienced, i.e., $\eta \leq 1$ and $f > f_{max}$. This is the worst possible state since reliability is low, congestion is experienced and energy is wasted. Therefore ESRT reduces reporting frequency aggressively in order to bring the network to state **OOR** as soon as possible. Note that reliability is a non-linear function of reporting frequency in state **(C,LR)** as shown in Figure 101. Hence in order to assure sufficient decrease in the reporting frequency, it is exponentially decreased and the new frequency is expressed by

$$f_{i+1} = f_i^{(\eta_i/k)} \tag{84}$$

where k denotes the number of successive decision intervals for which the network has remained in state **(C,LR)** including the current decision interval, i.e., $k \geq 1$. The aim is to decrease f with greater aggression if a state transition is not detected. Such a policy also ensures convergence for $\eta = 1$ in state **(C,LR)**.

5. **OOR** *(Optimal Operating Region)* : In this state, the network is operating within ϵ tolerance of the optimal point, where the required reliability is attained with minimum energy expenditure. Hence, the reporting frequency of source nodes is left unchanged for the next decision interval.

$$f_{i+1} = f_i \tag{85}$$

The entire ESRT protocol operation is summarized in the pseudo-algorithm given in Figure 103

```
    k = 1;
    ESRT()
      If (CONGESTION)
        If (η < 1)
        /* State=(C,LR) */
        /* Decrease Reporting Frequency Aggressively */
          f = f^{η/k};
          k = k + 1;
        else if (η > 1)
        /* State=(C,HR) */
        /* Decrease Reporting Frequency to Relieve Congestion; No Compromise on
Reliability */
          k = 1;
          f = f/η;
        end;
      else if (NO_CONGESTION)
        k = 1;
        If (η < 1 − ε)
        /* State=(NC,LR) */
        /* Increase Reporting Frequency Aggressively */
          f = f/η;
        else if (η > 1 + ε)
        /* State=(NC,HR) */
        /* Decrease Reporting Frequency Cautiously */
          f = f/2 (1 + 1/η);
        end;
        else if (1 − ε ≤ η ≤ 1 + ε)
        /* Optimal Operating Region */
        /* Hold Reporting Frequency */
          f = f;
        end;
    end;
```

Figure 103: Algorithm of the ESRT protocol operation.

7.4.2 Congestion Detection

In order to determine the current network state $\mathbf{S}_i$ in ESRT, the sink must be able to detect congestion in the network. However the conventional ACK/NACK-based detection methods for end-to-end congestion control purposes cannot be applied here. The reason once again lies in the notion of event-to-sink reliability rather than end-to-end reliability. Only the sink, and not any of the sensor nodes, can determine the reliability indicator η_i and act accordingly. Moreover, end-to-end retransmissions and ACK/NACK overheads are a waste of limited sensor resources. Hence, ESRT uses a congestion detection mechanism based on local buffer level monitoring in sensor nodes. Any sensor node whose routing

buffer overflows due to excessive incoming packets is said to be congested and it informs the sink of the same. The details of this mechanism are as follows.

In the event-to-sink model, the traffic generated during each reporting period, i.e., $1/f$, mainly depends on the reporting frequency f and the number of source nodes n. The reporting frequency f does not change within one reporting period since it is controlled periodically by the sink at the end of each decision interval with period of $\tau > 1/f$. Assuming n does not significantly change within one reporting period, the traffic generated during the next reporting period will have negligible variation. Therefore the amount of incoming traffic to any sensor node in consecutive reporting intervals is assumed to stay constant. This, in turn, signifies that the increment in the buffer fullness level at the end of each reporting interval is expected to be constant.

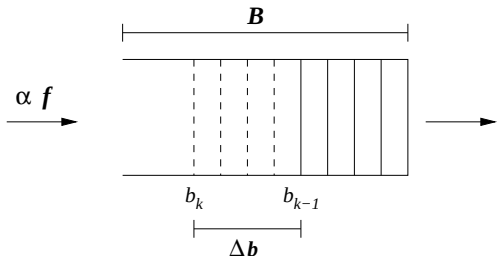

Figure 104: An illustration of buffer level monitoring in sensor nodes.

Let b_k and b_{k-1} be the buffer fullness levels at the end of k^{th} and $(k-1)^{th}$ reporting intervals respectively and B be the buffer size as in Figure 104. For a given sensor node, let Δb be the buffer length increment observed at the end of last reporting period, i.e.,

$$\Delta b = b_k - b_{k-1} \tag{86}$$

Thus if the sum of current buffer level at the end of k^{th} reporting interval and the last experienced buffer length increment exceeds the buffer size, i.e., $b_k + \Delta b > B$, the sensor node infers that it is going to experience congestion in the next reporting interval. Hence it sets the CN (Congestion Notification) bit in the header of the packets it transmits as shown in Figure 105. This notifies the sink for the upcoming congestion condition to be experienced in next reporting interval.

Hence if the sink receives packets whose CN bit is marked, then it infers that congestion

Event ID	*CN* (1 bit)	Destination	Time Stamp	Payload	FEC

Figure 105: A typical data packet with congestion notification field, which is marked to alert the sink for congestion.

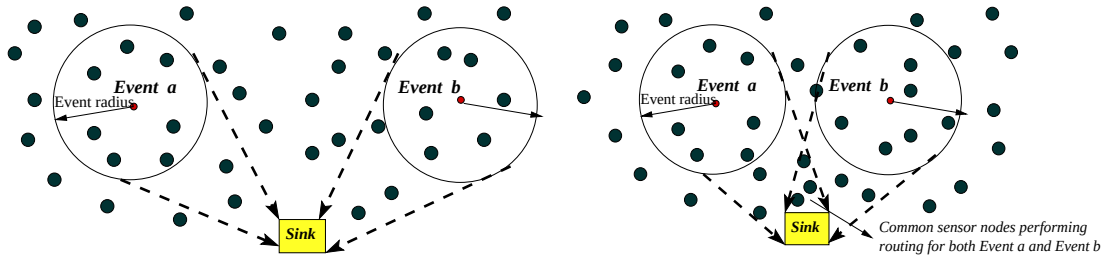

Figure 106: The multiple event occurrences in the same wireless sensor field (a) the flows generated by two events, i.e., *Event a* and *Event b*, are isolated (b) the flows pass through some common sensor nodes.

is experienced in the last decision interval. In conjunction with the reliability indicator η_i, the sink can now determine the current network state $\mathbf{S}_i$ at the end of decision interval i and act according to the rules in Section 7.4.1.

7.5 Multiple Event Occurrences

The ESRT protocol operation defined in Section 7.4 directly applies to the scenarios where a single event occurs in the wireless sensor field. In this section, ESRT protocol is extended in order to accommodate the cases where multiple events concurrently occur in the same wireless sensor field. In Section 7.5.1, how ESRT mechanisms can accurately detect multiple event occurrences and extract the required information for the protocol operation are explained. Then, the ESRT protocol operation in multiple event scenarios are presented in Section 7.5.2.

7.5.1 Multiple Event Detection

In order to address the scenarios where multiple events occur simultaneously, it is necessary to accurately obtain the following information:

1. Is there a single event or multiple concurrent events in the wireless sensor field?

2. If there are multiple events, are the generated data flows from sensor nodes to the sink passing through any common node?

In order to accurately capture the answers to these two questions, the sink utilizes the *Event ID* field of a data packet shown in Figure 105. Note that this field accurately provides the answer to the first question above. If all of the data packets received by the sink carry the same Event ID, then there is a single event occurrence in the wireless sensor field as shown in Figure 98. In this case, the sink achieves the desired event-to-sink reliability with minimum energy expenditure using the ESRT protocol operation shown in Figure 103 as explained in Section 7.4.

If the sink receives data packets carrying different event IDs in their Event ID fiels as shown in Figure 105, it infers that multiple concurrent events occured in the sensor field.

In this case, it is necessary to find the answer to the second question above, i.e., if there are any common sensor nodes serving as a router for the flows generated by these multiple events. This information is detrimental to the selection of appropriate ESRT operation due to the reasons as follows. If there is no common wireless sensor node performing routing for these multiple events occurred simultaneously, then the flows generated by these multiple events are isolated, i.e., do not share any common path as shown in Figure 106(a). Thus, in this case, ESRT protocol can address the event-to-sink reliability requirements of these multiple events individually with the default ESRT operation explained in Section 7.4.

If there exist common sensor nodes performing routing for the multiple events occurred simultaneously as shown in Figure 106(b), then the flows generated by these events are not isolated. In this case, treating them individually may not always lead to the best possible solution. This is because any action taken by the sink on any of these flows may alter the reliability level and the congestion situation of the other event flows. Therefore, protocol actions need to be taken cautiously and considering all of the concurrent event flows in the wireless sensor field. The updated ESRT protocol operation in order to accommodate these cases are explained in Section 7.5.2.

Hence, in order to determine the necessary protocol operation, the sink must accurately detect whether the flows generated by these multiple events pass through any common

sensor node functioning as a router. Furthermore, if indeed there exist such common router sensor nodes, it is necessary to learn which event flows share these common nodes. For this purpose, the sink utilizes the *Event ID* field of a data packet shown in Figure 105. Here, it is assumed that Event ID field shown in Figure 105 is a multidimensional field which can accommodate the Event IDs of several events occurring simultaneously. Therefore, the additional functionality required at the sensor nodes which perform routing can be stated as follows:

1. A sensor node keeps the *event-list*, i.e., the list of IDs of the events it serves as a router node in the wireless sensor field.

2. When the node receives a new data packet, it checks its event-list and the multidimensional Event ID field of this data packet.

   (a) If there exists an ID in its *event-list*, which is not in the multidimensional Event ID field of this data packet, the sensor node

   - adds this ID on top of the Event ID field of this data packet,
   - forwards the packet.

   (b) If there is not such ID, then the sensor node checks if its event-list includes the first element of the multidimensional Event ID field of this packet. If so, then the router sensor node leaves its event-list and the packet header intact and forwards the packet. If not, it adds the first element of the multidimensional Event ID field of this packet into its event-list and leaves the packet intact and forwards it.

To illustrate the accurate detection of a multiple events case, assume that a sensor node performs routing for the data packets generated by Events with Event IDs a and b as shown in Figure 106(b). Thus, this sensor node knows that it is indeed serving as a router node for the events a and b hence it has a and b in its *event-list*. Now, suppose that a data packet with only c in its Event ID field arrives at this sensor node. Hence, this sensor nodes adds a and b in the event ID field of the data packet and then forwards it. The sensor

node also updates its event-list since now it received a data packet generated by the Event c. Consequently, when the sink receives this data packet carrying c, a, and b in its Event ID field, it infers that the flows generated by the Events a, b, and c are not isolated and pass through common nodes. Accordingly, it performs the necessary protocol actions as explained in Section 7.5.2.

Note that the multiple event detection mechanism described above does not affect the accurate identification of the events by using the Event ID field in the packet headers. In fact, the first element of the multidimensional Event ID field is the ID of the event which originally generated the data packet. Hence, as the sink receives a data packet whose Event ID field carry multiple event IDs, it uses the first element of the multidimensional Event ID field to associate this data packet to the event in order to accurately calculate the observed event reliability as described in Section 7.4. Furthermore, note also that the mechanism described above requires only a simple lookup function as the additional functionality at the sensor nodes, and exploits the collective identification of the sensor nodes and avoids the need for individual sensor IDs.

7.5.2 ESRT Operation in Multiple Event Scenarios

As described in Section 7.5.1, the sink utilizes the *Event ID* field of a data packet in order to capture information about the multiple event occurrence in the wireless sensor field.

If a single event occurs in the wireless sensor field as shown in Figure 98, i.e., all of the data packets received by the sink carry the same Event ID, then the sink brings the network state **S** to the optimal operating region **OOR** with the default ESRT protocol operation as explained in Section 7.4.

For the multiple event occurrence scenarios, the ESRT protocol operation varies based on whether the flows generated by these multiple events are isolated or not as explained in Section 7.5.1. Hence, the detailed protocol operation for these two distinct cases are explained in the following sections.

7.5.2.1 Multiple Isolated Events

If there are multiple concurrent events in the sensor field, i.e., the sink receives data packets with different Event IDs, then the sink checks the Event ID fields of the data packets it received at the end of decision interval i. If all of the data packets have a single value in their multidimensional Event ID fields, it infers that the flows generated by these multiple events are isolated and do not share any common router sensor node as shown in Figure 106(a).

In this case, let $\mathbf{S}_i^k$ and f_i^k be the current network state and the reporting frequency for the event k. Note that ESRT determines the current network state for event k, i.e., $\mathbf{S}_i^k$, from the reliability indicator η_i^k computed by the sink for decision interval i as explained in Section 7.4. Thus, the sink calculates the updated reporting frequency f_{i+1}^k based on $\mathbf{S}_i^k$, η_i^k, and f_i^k and broadcasts it to the sensor nodes in the event radius of event k in order to bring the network state to the optimal operating region **OOR** for the flows generated by event k. Consequently, the sink achieves the event-to-sink reliability requirements of these multiple events individually with the default ESRT operation explained in Section 7.4.

7.5.2.2 Multiple Events Passing Through Common Nodes

If there are data packets which carry multiple event IDs in their Event ID fields, then the sink infers that there exist common sensor nodes routing the flows generated by these different events as shown in Figure 106(b). Therefore, the flows generated by these multiple events are not isolated. Hence, an action taken by the sink for any of these events may affect the reliability and congestion situation of the other events' flows.

In this case, instead of treating these event flows independently, it is better to take action cautiously and considering all of the concurrent event flows in the wireless sensor field. This is mainly because of the fact that the primary objective of ESRT is to achieve event-to-sink reliable transport. This leads to the fact that the event flows which are in different network states pose different levels of urgency in terms of protocol action. For example, while in state **(NC,HR)** no congestion is experienced and the observed reliability is higher than required, it is completely opposite in state **(C,LR)** where there is a congestion in the

network and the event-to-sink reliability is not achieved as shown in Figure 101. Hence, the event flows whose current network state are **(C,LR)** have greater urgency and hence high priority in terms of action to be taken by the sink. Similarly, although there is no congestion in both of the states **(NC,LR)** and **(NC,HR)**, the event flows which are currently in state **(NC,LR)** do not receive their desired reliability levels and has higher priority than the ones in state **(NC,HR)**. With this respect, the network states **{(C,LR), (NC,LR), (C,HR), (NC,HR)}** are grouped into *high priority states*, i.e., **(C,LR), (NC,LR)**, and *low priority states*, i.e., **(C,HR), (NC,HR)**, based on the observed reliability level associated with each of these network states.

Consequently, the sink takes the required action based on the priority of the network states of the multiple concurrent events sharing the same router sensor nodes. Let N_e be the number of concurrent events whose flows are passing through common router sensor nodes. The IDs of these events are obtained from the multidimensional Event ID field of the received data packets as explained in Section 7.5.1. Let $\mathbf{S}_i^k$ and f_i^k be the current network state and the reporting frequency for the event k for $k \in N_e$.

1. The sink determines the network state $\mathbf{S}_i^k$ for each of the flows generated by the event $k \in N_e$ at the end of decision interval i as described in Section 7.4.

2. If there are events whose network state are high priority, i.e., $\exists j \in N_e$ such that $\mathbf{S}_i^j$=**(C,LR)** or $\mathbf{S}_i^j$=**(NC,LR)**:

   (a) The sink immediately performs the default ESRT operation described in Section 7.4 for these events. That is, the sink calculates and broadcasts the updated reporting frequency f_{i+1}^j to the sensor nodes which are in the radius of event j, i.e., $\forall j$ with $\mathbf{S}_i^j$=**(C,LR)** or $\mathbf{S}_i^j$=**(NC,LR)**.

   This action is more urgent to take because these events are not reliably communicated to the sink hence the first priority action is to make these events reach their desired reliability levels.

   (b) The sink does not update the reporting frequencies for the other event flows whose network states are low priority, i.e., $f_{i+1}^j = f_i^j \ \forall j$ with $\mathbf{S}_i^j$=**(C,HR)** or

$\mathbf{S}_i^j$=**(NC,HR)**.

This is because the actions taken for the events flows whose network states are high priority (step 2.(a)) may affect these events which already have higher reliability. Therefore, any further simultaneous action to minimize energy expenditure of these flows is avoided to not to compromise their reliability levels. Note that this is also consistent with the primary objective of ESRT protocol operation which is to achieve event-to-sink reliability.

3. If there are no events whose network state are high priority, i.e., $\mathbf{S}_i^j$=**(C,HR)** or $\mathbf{S}_i^j$=**(NC,HR)** $\forall j \in N_e$, then the sink follows the default ESRT operation described in Section 7.4 for these events. That is, it calculates the updated reporting frequency f_{i+1}^j and broadcasts it to the sensor nodes which are in the event radius of event j $\forall j \in N_e$.

The sink repeats these steps until all of the event flows reach to the optimal operating region **OOR** as described in Section 7.4. As a result, the ESRT protocol operation described in Section 7.4 can accommodate the scenarios where multiple events occur simultaneously in the wireless sensor field.

7.6 ESRT Performance

In this section, both analytical and simulation results on the performance of ESRT protocol are presented. The results show that ESRT converges to state **OOR** starting from any of the other four initial network states $\mathbf{S}_i \in$**{(NC,LR), (NC,HR),(C,HR),(C,LR)}**. ESRT is self-configuring in this sense and can hence perform efficiently under random, dynamic topology frequently encountered in WSN applications.

The convergence times presented in this section are derived under the assumption that the r vs.f characteristic does not change appreciably within this duration. They can hence be interpreted as achievable lower bounds.

7.6.1 Analytical Results

First, some analytical results on ESRT performance are presented depending on the initial network state $\mathbf{S}_0$. Note that these results are obtained for the cases where a single event occurs in the sensor field although they may still apply for most of the multiple event cases. Recall that ESRT aims to reach state **OOR** starting from any initial state $\mathbf{S}_0$.

Lemma 1 *Starting from* $\boldsymbol{S}_0$*=**(NC,HR)**, and with linear reliability (η) behavior when the network is not congested, the network state remains unchanged until ESRT converges to state **OOR**.*

Proof 2 *The linear reliability (η) behavior for $f < f_{max}$ can be expressed as $f = \alpha\eta$, where α denotes the slope. ESRT conservatively decrements f as follows (equation (82))*

$$f_{i+1} = \frac{f_i}{2}\left(1 + \frac{1}{\eta_i}\right) \tag{87}$$

Hence,

$$\eta_{i+1} = \frac{1 + \eta_i}{2} \tag{88}$$

*Since $f_{i+1} < f_i$ from (87), it follows that $\boldsymbol{S}_i \in$ **{(NC,HR), (NC,LR),OOR}**, $\forall i \geq 0$ until ESRT converges. If possible, let $\boldsymbol{S}_{i+1}$=**(NC,LR)** when $\boldsymbol{S}_i$=**(NC,HR)** for some $i \geq 0$ before ESRT converges. Then,*

$$\eta_{i+1} = \frac{1 + \eta_i}{2} < 1 - \epsilon \tag{89}$$

*This implies that $\eta_i < 1 - 2\epsilon$, but $\eta_i > 1 + \epsilon$ since $\boldsymbol{S}_i$=**(NC,HR)**. Hence, $\boldsymbol{S}_i \neq$**(NC,LR)** for any $i \geq 0$ until ESRT converges. In conjunction with the earlier inference, it is concluded that $\boldsymbol{S}_i$=**(NC,HR)** $\forall i \geq 0$, until ESRT converges to state **OOR**.*

Lemma 3 *Starting from* $\boldsymbol{S}_0$*=**(NC,HR)**, and with linear reliability (η) behavior when the network is not congested, ESRT converges to state **OOR** in $\tau\lceil log_2\left(\frac{\eta_0 - 1}{\epsilon}\right)\rceil$ time units, where τ is the duration of the decision interval.*

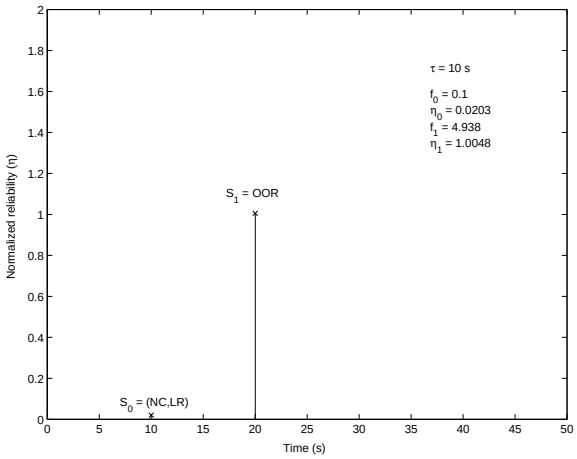

Figure 107: The ESRT protocol trace for $\mathbf{S}_0$=**(NC,LR)**. Convergence is attained in a total of two decision intervals. The trace values and states are also shown in the figure.

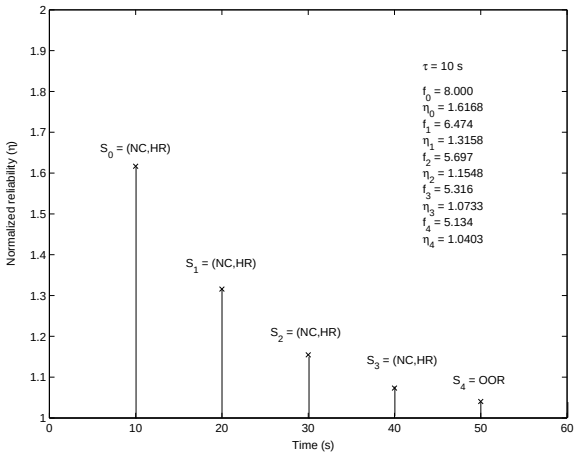

Figure 108: The ESRT protocol trace for $\mathbf{S}_0$=**(NC,HR)**. Convergence is attained in a total of five decision intervals. The trace values and states are also shown in the figure.

Proof 4 *To establish the convergence time, the proof proceeds as follows. Let the* j^{th} *decision interval be the first one where* $\boldsymbol{S_j}$**=OOR**. *It follows from Lemma 1 that* j *is the least index such that* $\eta_j < 1 + \epsilon$. *Using equation (88),*

$$\begin{array}{rcccl} \eta_j & = & \frac{\eta_{j-1}+1}{2} & < & 1+\epsilon \\ \eta_{j-1} & = & \frac{\eta_{j-2}+1}{2} & < & 1+2\epsilon \\ \vdots & & & & \\ \eta_1 & = & \frac{\eta_0+1}{2} & < & 1+2^{j-1}\epsilon \end{array} \tag{90}$$

Hence, $j > log_2\left(\frac{\eta_0-1}{\epsilon}\right)$ *and the result follows. Note that this represents the time required to reach state* **OOR** *in order to conserve maximum energy. The primary objective of reliable*

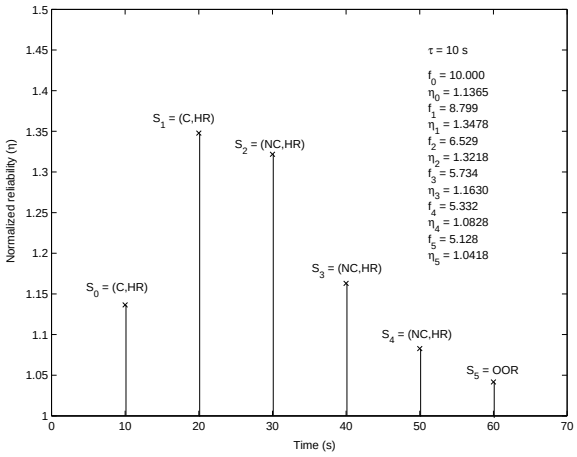

Figure 109: The ESRT protocol trace for $\mathbf{S}_0$=**(C,HR)**. Convergence is attained in a total of six decision intervals in this case. The trace values and states are also shown in the figure.

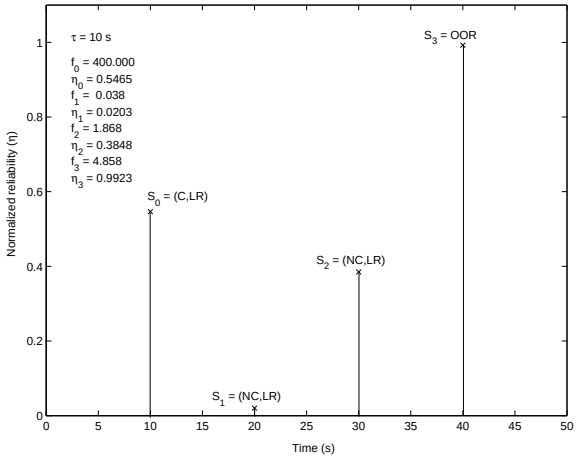

Figure 110: The ESRT protocol trace for $\mathbf{S}_0$=**(C,LR)**. Convergence is attained in a total of four decision intervals in this case. The trace values and states are also shown in the figure.

event detection is maintained all along by virtue of the conservative decrease (equation (87)).

Lemma 5 *With linear reliability (η) behavior when the network is not congested, the network state transition* $\boldsymbol{S_i}$**= (C,HR)**$\rightarrow \boldsymbol{S_{i+1}}$**=(NC,LR)** *is not possible for any* $i \geq 0$.

Proof 6 *The linear reliability (η) behavior for $f < f_{max}$ can be expressed as $f = \alpha\eta$, where α denotes the slope. It is seen from the r vs. f characteristics in Figs. 99, 100, and 101, that for every $f > f_{max}$ in state* **(C,HR)**, *there exists one $f' < f_{max}$ (in linear region) such that $\eta(f) = \eta(f')$.*

The proof now proceeds by contradiction. Let us assume that $\boldsymbol{S_{i+1}}$**=(NC,LR)** *when*

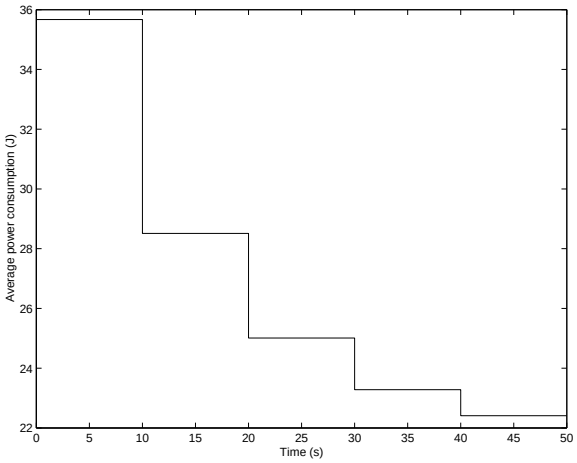

Figure 111: The average power consumption of sensor nodes in each decision interval for $\mathbf{S}_0$=**(NC,HR)**.

***S_i=(C,HR)**, for some $i \geq 0$. From the state definitions in Section 7.3.3 and update policy in Section 7.4.1, it follows that*

$$f_i' \frac{(1-\epsilon)}{\eta_i} > \frac{f_i}{\eta_i} \tag{91}$$

Hence, a necessary condition is

$$f_i' > \frac{f_i}{1-\epsilon} > f_i, \tag{92}$$

*but this is not true since $f_i > f_{max} > f_i'$. This completes the proof. In accordance with this result, there is no transition from state **(C,HR)** to **(NC,LR)** in the state diagram shown in Figure 102. This achieves the objective of relieving congestion and reducing energy consumption while not compromising on the event reliability (see Section 7.4.1).*

In order to determine the convergence times of the ESRT protocol starting from $\mathbf{S}_0$ $\in$ **{(C,HR),(C,LR)}**, the non-linear r vs. f behavior needs to be tracked analytically. However, this is beyond the present scope. Hence, the convergence in these two cases is studied using simulations.

7.6.2 Simulation Results

In order to study the convergence of ESRT using simulations, an evaluation environment is developed using *ns*-2 [111]. First, the simulation experiments are run for the scenario where a single event occurs in the wireless sensor field. The convergence results are shown

in Figs. 107 through 110 for initial network states $\mathbf{S}_0$=**(NC,LR)**,**(NC, HR)**,**(C,HR)**, and **(C,LR)**, respectively. The corresponding trace values (f_i, η_i) and states are listed within each figure. The energy conservation property of ESRT for $\mathbf{S}_0$=**(NC, HR)** is illustrated in Figure 111. For all the simulation results presented here, number of senders $n = 81$ and tolerance $\epsilon = 5\%$. The event radius was fixed at $40m$. Other simulation parameters are the same as those listed in Table 1 in Section 7.3.2.

It is seen from Figure 107 that the ESRT protocol for $\mathbf{S}_0$=**(NC, LR)** converges in a total of two decision intervals (2τ=20s). This is expected from the aggressive multiplicative policy employed. Lemmas 1, 2 and 3 in Section 7.6.1 can be verified from the trace values (f_i, η_i) and states listed within Figs. 108 and 109.

Furthermore, simulation experiments are run to assess the ESRT performance in the cases where multiple events occur simultaneously in the sensor field. Here, the number of intervals it takes for all of the event flows to converge to state **OOR**, and the average power consumption of the sensor nodes are observed. The simulation experiments are performed for varying number of multiple concurrent events.

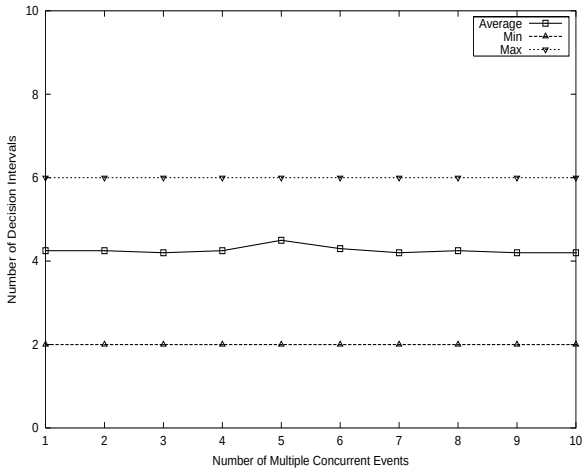

Figure 112: The number of decision intervals for all of the event flows to converge to state **OOR** for varying number of multiple concurrent events. In this set of experiments, the multiple concurrent events are isolated and their flows do not pass through any common router sensor node.

In the first scenario, simulation experiments are performed for the cases where the flows generated by the multiple events are isolated and do not share any common router

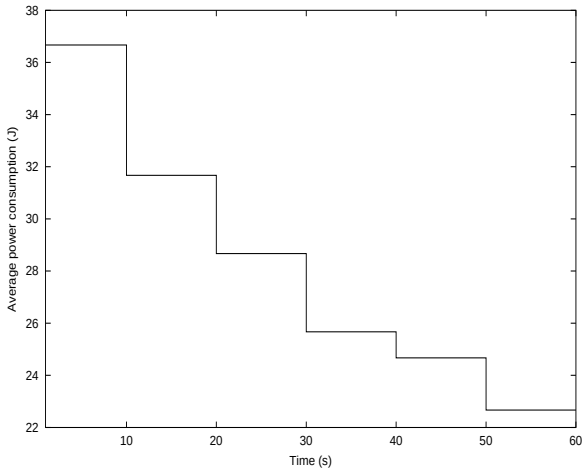

Figure 113: The average power consumption of sensor nodes in each decision interval for the case where 5 concurrent events occur in the wireless sensor field. In this case, the flows generated by these events are isolated.

Table 9: Summary of ESRT protocol operation in each of the five states.

Network State (S_i)	**Description**	**ESRT Action**
(NC,LR)	No Congestion, Low Reliability	Multiplicatively increase f
(NC,HR)	No Congestion, High Reliability	Decrease f conservatively
(C,HR)	Congestion, High Reliability	Decrease f aggressively
(C,LR)	Congestion, Low/equal Reliability	Decrease f exponentially
OOR	Optimal Operating Region	f remains unchanged

sensor node. As shown in Figure 112, the average number of decision intervals it takes for all of the event flows to converge to the state **OOR** does not vary significantly for varying number of multiple concurrent events. This is mainly because the flows generated by these multiple events are isolated and hence ESRT brings the network state of these flows to **OOR** individually as explained in Section 7.5.2. Note also that the minimum and maximum number of decision intervals required for convergence are 2 and 6, which are equal to the case where a single event occurs. Hence, the convergence to the **OOR** state is not delayed in the case of multiple isolated events.

Moreover, as shown in Figure 113, the average power consumed by the sensor nodes also show the same pattern observed for a single event scenario as shown in Figure 111. This is also because of the fact that the sink takes action for the flows generated by the multiple isolated events independently. Therefore, the average power consumption decreases with

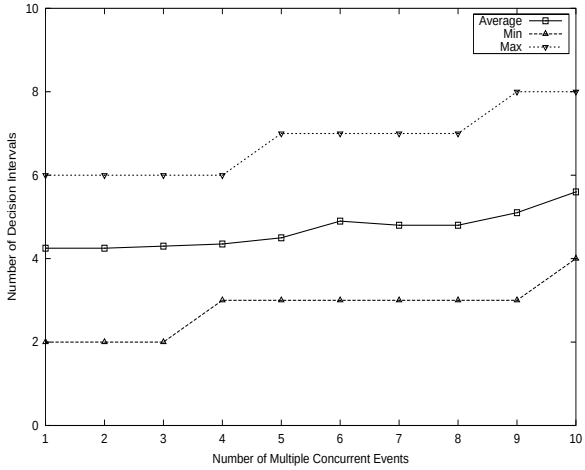

Figure 114: The number of decision intervals for all of the event flows to converge to state **OOR** for varying number of multiple concurrent events. In this set of experiments, the multiple concurrent events are not isolated.

time as the ESRT protocol works to minimize the energy expenditure while maintaining the event-to-sink reliability.

In the second scenario, simulation experiments are performed for the cases where the flows generated by the multiple events are not isolated and there are common router sensor nodes routing these multiple flows in the sensor field. As shown in Figure 114, the average number of decision intervals it takes for all of the event flows to converge state **OOR** slightly increases with the number of multiple concurrent events. This is mainly because the flows generated by these multiple events are not isolated and hence ESRT considers the priority of the current network states of these flows as explained in Section 7.5.2. Therefore, the sensor nodes which are in the radius of the events that already have adequate reliability may not experience reporting frequency update at the end of each decision interval. Consequently, the number of decision intervals it takes for those events to converge increases. Note also that the minimum and maximum number of decision intervals required for convergence also vary with the number of multiple concurrent events due to the same reason. However, as shown in Figure 114, the increase in the convergence time is very small even in case of 10 non-isolated concurrent events. Hence, the ESRT protocol can effectively address the cases where multiple events occur simultaneously.

Furthermore, as shown in Figure 115, the average power consumed by the sensor nodes

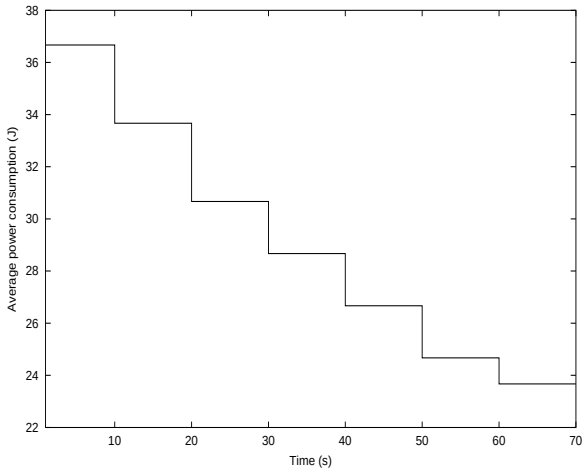

Figure 115: The average power consumption of sensor nodes in each decision interval for the case where 5 concurrent events occur in the wireless sensor field. In this case, the flows generated by these events are not isolated.

also show the same pattern observed for the previous case in Figure 113. However, the decrease in the average consumed power is slightly slower in this case. This is also because the fact that the sink may not take any action for some of the flows which already have adequate reliability levels. Note that this result is also consistent with the average convergence time results shown in Figure 114.

7.6.3 Suitable Choice of ϵ

For practical purposes, ESRT uses a tolerance zone of ϵ around the optimal operating point P_1 in Figure 101. If at the end of decision interval i, the reliability η_i is within [1-ϵ,1+ϵ] and if no congestion is detected in the network, then the network is in state **OOR**. The event is deemed to be reliably detected at the sink and the reporting frequency remains unchanged. Greater proximity to the optimal operating point can hence be achieved with small ϵ. However, as seen from Lemma 2 in Section 7.6.1, smaller the ϵ, greater the convergence time. Hence, a good choice of ϵ is one that balances the tolerance and convergence requirements. For example, a 1% tolerance requirement can offset the convergence time by as much as 7τ time units when $\mathbf{S}_0$=**(NC,HR)**. Note however that reliable event detection is maintained all along (Lemma 2 in Section 7.6.1) due to the conservative decrease.

CHAPTER VIII

CONCLUSIONS AND FUTURE RESEARCH DIRECTIONS

In this thesis, new advanced transport protocols have been developed for reliable data transport and real-time multimedia delivery in the next generation heterogeneous wireless network architectures, i.e., Next Generation Wireless Internet, InterPlaNetary Internet, and Wireless Sensor Networks. The following six areas have been investigated under this research and each of them is described in the following subsections:

1. Analytical Rate Control in Hybrid Wired/Wireless Networks
2. Multimedia Rate Control in Satellite Networks
3. Adaptive Transport in NGWI
4. Reliable Data Transport in IPN Internet
5. Integrated Transmission in IPN Internet
6. Reliable Event Transport in WSN

8.1 Research Contributions

8.1.1 Analytical Rate Control in Hybrid Wired/Wireless Networks

Next Generation Wireless Internet (NGWI) is expected to provide a wide range of services including real-time multimedia to mobile users. The real-time multimedia traffic transport requires rate control deployment to protect shared Internet from unfairness and further congestion collapse. However, the existing solutions are mostly for the wired Internet and hence they do not address the challenges in the wireless environments.

In Chapter 2, a new analytical rate control protocol was introduced for real-time multimedia traffic over wireless networks. ARC is an end-to-end analytical rate control protocol

which is designed to produce TCP-friendly real-time traffic flows while maintaining high throughput performance in wireless networks. The desired TCP behavior over wireless links is captured based on the end-to-end path model developed. The asymptotic throughput equation is derived and used as the control equation for the ARC rate control scheme. Its equation-based approach provides basis for multimedia traffic support in terms of time-constraint such as delay and/or jitter bounds.

The performance evaluation of ARC protocol were performed with simulation experiments. The results showed that ARC significantly improves the rate control throughput performance over hybrid wired/wireless paths and provides multimedia traffic support without penalizing TCP sources sharing same path.

8.1.2 Multimedia Rate Control in Satellite Networks

In Chapter 3, a new rate control scheme (RCS) was developed for real time traffic in networks with high bandwidth-delay products and high link error rates such as satellite networks. RCS improves throughput using dummy packets to probe available network resources. Dummy packets are low priority packets which are used to probe the link resources and hence help source to distinguish packet losses due to link errors and congestion. Therefore, RCS requires the routers along the connection to implement some priority mechanism. The main feature of RCS is that it is an end-to-end protocol, i.e., it needs to be implemented only at the source and destination. RCS is a TCP-friendly rate control scheme, which halves its transmission rate in case of packet loss. RCS can then resume its original rate very rapidly in case of packet losses due to link errors. If the packet loss is due to congestion, RCS follows TCP-friendly behavior rules and increases transmission rate additively after rate halving. For temporal signal loss situations, RCS avoids unnecessary rate throttles and resumes its original transmission rate very rapidly after signal is back. Simulation results showed that RCS achieves high performance in terms of throughput, fairness and real-time multimedia in networks with high bandwidth-delay product and high link error rate while providing TCP-friendly behavior.

8.1.3 Adaptive Transport Layer for NGWI

The transport layer protocols developed for different wireless architectures do not provide a single solution to address the heterogeneities posed by the Next Generation Wireless Internet (NGWI). The inadequacies of the current transport layer solutions necessitates a new transport layer that can address these heterogeneities. There has been no single proposed transport layer solution to serve the NGWI objective.

In Chapter 4, a new unified adaptive transport layer (ATL) was presented to realize the NGWI objective. ATL incorporates a new adaptive transport protocol, TCP-ATL, for reliable data transport; and a new adaptive AIMD rate control protocol, RCP-ATL, for multimedia delivery in the NGWI. Both TCP-ATL and RCP-ATL deploy new adaptive congestion/rate control methods that dynamically adapt their AIMD control parameters to the current wireless link conditions to maintain high throughput performance and to provide multimedia traffic support in NGWI. The experiment results showed that ATL protocols can maintain high performance throughout the different wireless architectures. Their adaptive AIMD configuration methods also help avoid performance degradation for high packet loss rates. Furthermore, since the developed AIMD parameter adjustment strategy takes fairness into consideration, ATL protocols also maintain fairness to the wired TCP sources sharing the same bottleneck. Moreover, ATL protocols do not bring any additional overhead and can be developed on top of any of the existing TCP or AIMD rate control protocols.

The experiment results showed that both ATL protocols achieve very high performance in the heterogeneous wireless architectures for wide ranges of packet loss probability and link delay. TCP-ATL is shown to significantly improve the throughput performance over Snoop (with TCP-Sack) [20] and TCP-Westwood [31] for WLAN environments; WTCP [99] and TCP-Westwood [31] for wide-area 3G cellular, TCP-Peach+ [10], [11] and TCP-Westwood [31] for satellites environments. Hence, instead of using different transport protocols developed for specific architectures, TCP-ATL achieves high throughput performance in all of the three different wireless architectures by adapting its protocol configuration. TCP-ATL is also shown to address the blackout situations and maintain its throughput performance.

Both TCP-ATL and RCP-ATL are shown to preserve fairness to the wired TCP sources sharing the same bottleneck. RCP-ATL is also shown to improve jitter performance in wireless environments. As a result, ATL protocol suite addresses the challenges posed by the NGWI and significantly improves the performance for reliable data and multimedia transport in NGWI.

8.1.4 Reliable Data Transport in IPN Internet

The inadequacy of the current TCP protocols in Interplanetary Backbone Network has already been known and the need for new transport protocol has been pointed out in [2]. In order to address this need, a new reliable transport protocol, TP-Planet, was developed in Chapter 5. The objective of TP-Planet is to address the challenges posed by the InterPlaNetary Backbone Network for reliable data delivery and achieve high throughput performance. TP-Planet deploys a rate-based additive-increase multiplicative-decrease (AIMD) congestion control scheme. It runs on top of Internet Protocol (IP) layer and does not require any specific modification to the lower layers in the current TCP/IP suite. Performance evaluation via simulation experiments revealed that TP-Planet significantly improves the throughput performance and addresses the challenges posed by deep space communication networks.

TP-Planet replaces the inefficient slow start algorithms with a novel Initial State algorithm, which captures link resources in a very fast and controlled manner. Simulation experiments showed that TP-Planet can reach high initial data rates very quickly. In order to address the challenges due to extremely high propagation delay, TP-Planet deploys a rate-based additive-increase multiplicative-decrease (AIMD) congestion control, whose AIMD parameters are tuned to help avoid throughput degradation. A new congestion control mechanism, which decouples congestion decision from single packet loss events, is developed to minimize the erroneous congestion decisions due to high link errors. Consequently, TP-Planet improves throughput with a factor more than 10^3 compared to the current TCP protocols. In order to reduce the effects of blackout conditions on the throughput performance, TP-Planet incorporates Blackout State behavior into the protocol operation.

By this way, it achieves up to 14% performance improvement in blackout conditions. The bandwidth asymmetry problem is addressed by the adoption of delayed SACK options. As a result, TP-Planet is a reliable transport protocol equipped with diverse set of algorithms and functionalities, which can address the requirements of the InterPlaNetary Backbone Network.

Note that TP-Planet is mainly implemented at the Interplanetary Backbone Network nodes, i.e., the TP-Planet source and sink are the ground station gateway at the Earth and the planetary gateway connected to the relay satellites orbiting around the outer-space planets. The end-to-end transport control can be achieved by using the existing transport protocols developed for terrestrial wireless networks on the PlaNetary Surface Networks in conjunction with TP-Planet on the InterPlaNetary Backbone Network. However, the detailed description of such cooperation and its performance evaluation are beyond the scope of this work and left for future study.

8.1.5 Integrated Transmission in IPN Internet

The conventional end-to-end transport layer solutions are not suitable for IPN Internet paths mainly characterized by intermittent connectivity, high link error rates, extremely high propagation delays, and bandwidth asymmetry. In Chapter 6, a new Integrated Transmission Protocol (ITP) was presented for reliable data transport in the IPN Internet. ITP is a unified transmission protocol solution for hop-by-hop congestion control and reliability mechanisms specifically tailored for IPN Internet links with intermittent connectivity. Exploiting the hop-by-hop nature of the connections, ITP unifies the common functionalities of the conventional transport and link layers. The objective of ITP is to address the challenges and to achieve high performance reliable data transport on deep space links of the IPN Internet. ITP deploys a new rate-based hop-by-hop local flow control (LFC) mechanism; which exploits the local resource availability and traffic information at the receiver in order to provide explicit available bandwidth feedback to the sender. LFC mechanism decouples congestion decision from reliability to avoid the erroneous congestion decisions due to high link errors and avoids the delayed-feedback problem. ITP incorporates the

selective-acknowledgment (SACK) based Automatic Repeat reQuest (ARQ) to assure hop-by-hop local packet-level reliability. To reduce the effects of blackout conditions on the performance, ITP includes the Blackout Mitigation (BM) procedure. The optimum packet size is analytically obtained to further improve the transmission efficiency over deep space links. Bandwidth asymmetry problem is addressed by the adoption of delayed SACK.

It has been shown via simulation experiments that the ITP significantly improves the throughput performance and addresses the challenges posed by the IPN Internet. Throughput performance of ITP is also shown to be not affected by the bandwidth asymmetry problem as LFC mechanism does not depend on ACKs for data rate determination. Note also that unlike the existing end-to-end solutions, ITP can be implemented on any IPN node which involve in hop-by-hop custodial bundle transport.

8.1.6 Event-to-Sink Reliable Transport in WSN

The notion of event-to-sink reliability is necessary for reliable transport of event features in WSN. This is due to the fact that the sink is only interested in the collective information of a number of source nodes and not in individual sensor reports. This is also the reason why traditional end-to-end reliability notions and transport solutions are inappropriate for WSN. Based on such a collective reliability notion, a new reliable transport scheme for WSN, the event-sink reliable transport (ESRT) protocol, was presented in Chapter 7.

ESRT is a novel transport solution developed to achieve reliable event detection with minimum energy expenditure and congestion resolution functionality. This is the first study of reliable transport in WSN from the event-to-sink perspective.

ESRT has been tailored to meet the unique requirements of WSN. Its congestion control component serves the dual purpose of achieving reliability and conserving energy. The algorithms of ESRT mainly run on the sink and require minimal functionality at resource constrained sensor nodes. The primary objective of ESRT is to configure the network as close as possible to the optimal operating point, where the required reliability is achieved with minimum energy consumption and without network congestion. Thus,

ESRT protocol operation is determined by the current network state based on the reliability achieved and the congestion condition. In this regard, five possible network states $\mathbf{S}_i \in$ **{(NC,LR),(NC,HR),(C,HR),(C,LR),OOR}** were identified and ESRT operation in each of these states was discussed in detail in Section 7.4.1. The main ideas are summarized in Table 4.

ESRT protocol also accommodates the scenarios where multiple events concurrently occur in the wireless sensor field. The sink exploits the collective identification of the sensor nodes in order to accurately capture if a single or multiple events occur and in case of multiple events whether the flows generated by these events are isolated or not. Hence, according to this information, the sink uses the ESRT protocol to achieve the required event-to-sink reliability levels for each of these concurrent events.

Analytical performance evaluation and simulation results show that ESRT converges to state **OOR** regardless of the initial network state $\mathbf{S}_0$. Furthermore, the simulation experiments show that ESRT can also achieve the required event-to-sink reliability in case of multiple concurrent events. This self-configuring aspect of ESRT is valuable under random, dynamic topology frequently encountered in WSN applications.

8.2 Future Research Directions

- **Adaptive transport in proximity wireless environments:** The Next Generation Wireless Internet (NGWI) will converge a wide range of heterogeneous wireless network architectures from wireless LANs to satellite networks. In this thesis, the adaptive transport layer (ATL) suite was developed to address these heterogeneities. In addition to that, it may be required to integrate wireless sensor networks to the NGWI architecture. To achieve this, ATL mechanisms must be extended to address the challenges posed by the proximity wireless environments such as wireless ad hoc networks, wireless sensor networks. The new solution must also be adaptive to changing wireless architecture as well as able to address the individual architectural requirements.

- **Reliable transport in PlaNetary Networks:** The PlaNetary Networks, which

will compose scientific devices and communication nodes on and around the planets, will also require reliable data transport mechanisms for successful scientific delivery. Although the existing solutions proposed in the literature for terrestrial wireless networks and satellite networks may be applicable, their performance evaluation must be assessed to understand their potential shortcomings and consequently new solutions to be developed.

- **Spatio-temporal correlation in WSN:** In addition to the collaborative nature of WSN, the spatio-temporal correlation between the sensor observations is another significant and unique characteristic of WSN which can be exploited to drastically enhance the overall network performance by reducing the energy consumption. While the Event-to-Sink Reliable Transport protocol achieves reliable event detection with minimum energy expenditure, it can be extended to explicitly exploit the correlation in the WSN in order to achieve further performance improvements.

- **Integration of NGWI, IPN Internet, and WSN:** In this thesis, advanced transport layer protocols have been developed for NGWI, IPN Internet, and WSN. However, there will be scenarios where a connection path may go through all of these next generation heterogeneous wireless network architectures. For example, a sensor node which is part of a WSN on a planet surface may need to be reached from a node in the NGWI. Therefore, integration of the proposed transport layer protocols must be investigated in order to achieve seamless transport across the NGWI, IPN Internet and WSN.

REFERENCES

[1] A. A. Abouzeid, S. Roy, and M. Azizoglu, "Stochastic Modeling of TCP over Lossy Links," in *Proc. IEEE INFOCOM 2000*, Vol. 3, pp. 1724-1733, March 2000.

[2] O. B. Akan, J. Fang, I. F. Akyildiz, "Performance of TCP Protocols in Deep Space Communication Networks," *IEEE Commun. Lett.*, Vol. 6, No. 11, pp. 478-480, November 2002.

[3] O. B. Akan and I. F. Akyildiz, "ARC: The Analytical Rate Control Scheme for Real-Time Traffic in Wireless Networks," *to appear in IEEE/ACM Transactions on Networking,* June 2004.

[4] I. F. Akyildiz, O. B. Akan, and G. Morabito, "A Rate Control Scheme for Adaptive Real-Time Applications in IP Networks with Lossy Links and Long Round Trip Times," *to appear in IEEE/ACM Transactions on Networking,* 2004.

[5] O. B. Akan, I. F. Akyildiz, and Y. Sankarasubramaniam, "Event-to-Sink Reliable Transport in Wireless Sensor Networks," *to appear in in IEEE/ACM Transactions on Networking*, 2004.

[6] O. B. Akan, J. Fang, and I. F. Akyildiz, "TP-Planet: A Reliable Transport Protocol for InterPlaNetary Internet," *IEEE Journal on Selected Areas in Communications (JSAC)*, vol. 22, no. 2, pp. 348-361, February 2004.

[7] O. B. Akan and I. F. Akyildiz, "ATL: An Adaptive Transport Layer for Next Generation Wireless Internet," *to appear in IEEE Journal on Selected Areas in Communications (JSAC)*, 2nd Quarter 2004.

[8] O. B. Akan, "On the Throughput Analysis of Rate-Based and Window-Based Congestion Control Schemes," *Computer Networks Journal (Elsevier Science)*, vol. 44, no. 5, pp. 701-711, April 2004.

[9] O. B. Akan and I. F. Akyildiz, "Integrated Transmission Protocol for InterPlaNetary Backbone Network," *Submitted for publication*, April 2004.

[10] I. F. Akyildiz, G. Morabito, and S. Palazzo, "TCP-Peach: A New Congestion Control Scheme for Satellite IP Networks," *IEEE/ACM Trans. Networking,* Vol. 9, No. 3, pp. 307-321, June 2001.

[11] I. F. Akyildiz, X. Zhang, and J. Fang, "TCP-Peach+: Enhancement of TCP-Peach for Satellite IP Networks," *IEEE Commun. Lett.*, Vol. 6, No. 7, pp. 303-305, July 2002.

[12] I. F. Akyildiz, O. B. Akan, C. Chen, J. Fang, and W. Su, "InterPlaNetary Internet: State-of-the-Art and Research Challenges," *Computer Networks Journal (Elsevier Science)*, vol. 43, no. 2, pp. 75-112, October 2003.

[13] E. Altman, K. Avrachenkov, and C. Barakat, "TCP in presence of bursty losses," in *Proc. ACM SIGMETRICS 2000*, pp. 124-133, June 2000.

[14] S. Aramvith, I. M. Pao, and M. T. Sun, "A Rate-Control Scheme for Video transport over Wireless Channels," *IEEE Trans. Circuits Syst. Video Technol.*, Vol. 11, No. 5, pp. 569-580, May 2001.

[15] V. Arya and T. Turletti, "Accurate and Explicit Differentiation of Wireless and Congestion Losses," in *Proc. Workshop on Mobile and Wireless Networks (MWN)*, Providence, Rhode Island, May 2003.

[16] A. Bakre and B. R. Badrinath, "I-TCP: Indirect-TCP for Mobile Hosts," in *Proc. ICDCS 1995*, May 1995.

[17] H. Balakrishnan, V. N. Padmanabhan, S. Seshan, and R. H. Katz, "A Comparison of Mechanisms for Improving TCP Performance over Wireless Links," *IEEE/ACM Trans. Networking*, Vol. 5, No. 6, pp. 756-769, December 1997.

[18] H. Balakrishnan, H. S. Rahul, and S. Seshan, "An Integrated Congestion Management Architecture for Internet Hosts," in *Proc. ACM SIGCOMM 1999*, pp. 175-187, September 1999.

[19] H. Balakrishnan, V. N. Padmanabhan, R. H. Katz, "The Effects of Asymmetry on TCP Performance," *Proc. ACM MOBICOM 1997,* pp. 77-89, Hungary, September 1997.

[20] H. Balakrishnan, S. Seshan, E. Amir, and R. H. Katz, "Improving TCP/IP Performance over Wireless Networks," in *Proc. ACM MOBICOM 1995*, pp. 2-11, November 1995.

[21] D. Barman, I. Matta, "Effectiveness of Loss Labeling in Improving TCP Performance in Wired/Wireless Networks," in *Proc. IEEE ICNP 2002*, pp. 2-11, November 2002.

[22] K. Bhasin, J. Hayden, J. R. Agre, L. P. Clare, T. Y. Yan, "Advanced Communication and Networking Technologies for Mars Exploration," *19th Annual AIAA International Communications Satellite Systems Conference,* Toulouse, France, April 2001.

[23] K. Bhasin, J. Hayden, "Space Internet Architectures and Technologies for NASA Enterprises," *Proc. IEEE Aerospace 2001*, Vol. 2, pp. 931-941, 2001.

[24] G. Bianchi, A. Capone, and C. Petrioli, "Throughput Analysis of End-to-End Measurement-Based Admission Control in IP," in *Proc. IEEE INFOCOM 2000*, Vol. 3, pp. 1461-1470, March 2000.

[25] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An Architecture for Differentiated Services," *RFC 2474*, December 1998.

[26] J. C. Bolot and T. Turletti, "Experience with Rate Control Mechanisms for Packet Video in the Internet", *ACM SIGCOMM Computer Communication Review,* Vol. 28, No. 1, pp. 4-15, January 1998.

[27] J. Bolot, S. Parisis, and D. Towsley, "Adaptive FEC-based Error Control for Internet Telephony", in *Proc. IEEE INFOCOM 1999*, Vol. 3, pp. 1453-1460, March 1999

[28] L. S. Brakmo, S. O Malley, L. L. Peterson, "TCP Vegas: New Techniques for Congestion Detection and Avoidance," *Proc. ACM SIGCOMM 1994*, pp. 24-35, October 1994.

[29] L. Breslau, E. W. Knightly, S. Shenker, I. Stoica, and H. Zhang, "Endpoint Admission Control: Architecture Issues and Performance," in *Proc. ACM SIGCOMM 2000*, pp. 57-70, Stockholm, 2000.

[30] S. Burleigh, A. Hooke, L. Torgerson, K. Fall, V. Cerf, B. Durst, and K. Scott, "Delay-Tolerant Networking: An Approach to Interplanetary Internet," *IEEE Communications Magazine*, Vol. 41, No. 6, pp. 128-136, June 2003.

[31] C. Casetti, M. Gerla, S. Mascolo, M. Y. Sanadidi, and R. Wang, "TCP-Westwood: Bandwidth Estimation for Enhanced Transport over Wireless Links," in *Proc. ACM MOBICOM 2001*, pp. 287-297, Rome, July 2001.

[32] S. Cen, C. Pu, and J. Walpole, "Flow and Congestion Control for Internet Media Streaming Applications," in *Proc. SPIE Multimedia Computing and Networking*, pp. 250-264, January 1998.

[33] V. Cerf et. al., "Delay-Tolerant Network Architecture," *IETF Draft*, *¡draft-irtf-dtnrg-arch-01.txt¿*, October 2003.

[34] A. Chockalingam, M. Zorzi, L. B. Milstein, and P. Venkataram, "Performance of a Wireless Access Protocol on Correlated Rayleigh-Fading Channels with Capture," *IEEE Trans. Commun.*, Vol. 46, No. 5, pp. 644-655, May 1998.

[35] D. Chiu and R. Jain, "Analysis of the Increase and Decrease Algorithm for Congestion Avoidance in Computer Networks," *Computer Networks and ISDN Systems Journal*, Vol. 17, No. 1, pp. 1-14, June 1989.

[36] Cisco Systems' Web Page, available at *http://www.cisco.com*, July 2003.

[37] A. J. Cobb and P. Agrawal, "Congestion or Corruption? A Strategy for Efficient Wireless TCP Sessions," in *Proc. IEEE Symposium on Computers and Communications*, pp. 262-268, 1995.

[38] Consultative Committee for Space Data Systems, "Space Communications Protocol Specification-Transport Protocol (SCPS-TP)", *Recommendation for Space Data Systems Standards, CCSDS 714.0-B-1.*, Blue Book. Issue 1, Washington, D.C.: CCSDS, May 1999.

[39] Consultative Committee for Space Data Systems, "Telemetry Channel Coding", *Recommendation for Space Data System Standards, CCSDS 101.0-B-6.*, Blue Book. Issue 6, Washington, D.C.: CCSDS, October 2002.

[40] D. J. Costello, Jr., J. Hagenauer, H. Imai, and S. B. Wicker, "Applications of Error-Control Coding," *IEEE Trans. Info. Theory*, Vol. 44, No. 6, pp. 2531-2560, October 1998.

[41] S. Deering and R. Hinden, "Internet Protocol Version 6 (IPv6) Specification," *RFC 2460*, December 1998.

[42] S. Dolinar, D. Divsalar, and F. Pollara, "Turbo Codes and Space Communications," Communications Systems and Research Section, Jet Propulsion Laboratory, Califorina Institute of Technology.

[43] R. C. Durst, G. J. Miller, E. J. Travis, "TCP Extensions for Space Communications," *ACM/Kluwer Wireless Networks (WINET) Journal,* Vol. 3, No. 5, pp. 389-403, October 1997.

[44] R. C. Durst, P. D. Feighery, K. L. Scott, "Why not use the Standard Internet Suite for the Interplanetary Internet?," *http://www.ipnsig.org/techinfo.htm.*

[45] R. C. Durst, "Delay-Tolerant Networking: An Example Interplanetary Internet Bundle Transfer," *Internet Draft, draft-irtf-dtnrg-ipn-bundle-xfer-00.txt,* March 2003.

[46] T. Ferrari, W. Almesberger, and J. Y. Le Boudec, "SRP: A Scalable Resource Reservation Protocol for the Internet," *Computer Communications*, Vol. 21, No. 14, pp. 1200-1211, September 1998.

[47] S. Floyd, "TCP and Explicit Congestion Notification," *ACM Computer Communication Review,* Vol. 24, No. 5, pp. 10-23, October 1994.

[48] S. Floyd and K. Fall, "Promoting the Use of End-to-End Congestion Control in the Internet," *IEEE/ACM Trans. Networking*, Vol. 7, No. 4, pp. 458-472, August 1999.

[49] S. Floyd, M. Handley, and J. Padhye, "A Comparison of Equation-Based and AIMD Congestion Control," *ACIRI Technical Report* http://www.aciri.org/tfrc/aimd.pdf, May 2000.

[50] S. Floyd and V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance," *IEEE/ACM Trans. Networking*, Vol. 1, No. 4, pp. 397-413, August 1993.

[51] S. Floyd and T. Henderson, "The NewReno Modification to TCP's Fast Recovery Algorithm," *RFC 2585*, April 1999.

[52] S. Floyd, M. Handley, J. Padhye, and J. Widmer, "Equation-Based Congestion Control for Unicast Applications," in *Proc. ACM SIGCOMM 2000*, pp. 45-58, August 2000.

[53] S. Floyd, V. Jacobson, C. Liu, S. Macanne, and L. Zhang, "A Reliable Multicast Framework for Lightweight Sessions and Application Level Framing," *IEEE/ACM Trans. Networking,* Vol. 5, No. 6, pp.784-803, Dec. 1997.

[54] E. N. Gilbert, "Capacity of A Burst-Noise Channel," *Bell Syst. Tech. J.*, Vol. 39, pp. 1253-1265, September 1960.

[55] T. Goff, J. Moronski, D. S. Phatak, V. Gupta, "Freeze-TCP: A True End-to-End TCP Enhancement Mechanism for Mobile Environments," *Proc. IEEE INFOCOM 2000*, Vol. 3 , pp. 1537-1545, Israel, 2000.

[56] J. Heinanen, Telia Finland, F. Baker, W. Weiss, and J. Wroclawski, "Assured Forwarding PHB Group," *RFC 2597*, June 1999.

[57] T. R. Henderson, R. H. Katz, "Transport Protocols for Internet-Compatible Satellite Networks," *IEEE J. Select. Areas Commun.,* Vol. 17, No. 2, pp. 326-344, February 1999.

[58] H-Y. Hsieh and R. Sivakumar, "A Transport Layer Approach for Achieving Aggregate Bandwidths on Multi-homed Mobile Hosts," *Proc. ACM MOBICOM 2002*, pp. 83-94, Atlanta, Georgia, September 2002.

[59] IEEE 802.11, "Wireless LAN medium access control (MAC) and physical layer (phy) specification," 1999.

[60] H. Inamura, G. Montenegro, R. Ludwig, A. Gurtov, and F. Khafizov, "TCP over Second (2.5G) and Third (3G) Generation Wireless Networks," *IETF Internet-Draft*, draft-ietf-pilc-2.5g3g-12, December 2002.

[61] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks," in *Proc. ACM MOBICOM 2000,* pp. 56-67, Boston, Massachussetts, August 2002.

[62] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks," in *Proc. ACM MOBICOM 2000,* pp. 56-67, Boston, MA, USA, August 2000.

[63] S. Jacobs and A. Eleftheriadis, "Real-Time Dynamic Rate Shaping and Control for Internet Video Applications," in *Proc. The Workshop on Multimedia Signal Processing 1997*, June 1997.

[64] V. Jacobson, "Congestion Avoidance and Control," *Proc. ACM SIGCOMM*, pp. 314-329, August 1988.

[65] V. Jacobson, "Congestion Avoidance and Control," *Proc. ACM SIGCOMM 1988,* pp. 314-329, Stanford, August 1988.

[66] V. Jacobson, "Berkeley TCP Evolution from 4.3-Tahoe to 4.3-Reno," *Proc. British Columbia Internet Engineering Task Force*, July 1990.

[67] M. Jain, C. Dovrolis, "End-to-End Available Bandwidth: Measurement Methodology, Dynamics, and Relation with TCP Throughput," in *Proc. ACM SIGCOMM 2002*, pp. 295-308, August 2002.

[68] D. Johnson, D. Maltz, Y. Hu, and J. Jetcheva, "The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR)," *http://www.ietf.org/internet-drafts/draft-ietf-manet-dsr-07.txt*, IETF MANET working group, Internet draft, February 2002.

[69] H. Kanakia, P. Mishra, and A. Reibman, "An Adaptive Congestion Control Scheme for Real-Time Packet Video Transport," in *Proc. ACM SIGCOMM 1993*, pp. 20-31, San Francisco, September 1993.

[70] P. Karn and C. Partridge, "Improving Round-Trip Time Estimates in Reliable Transport Protocol," *ACM Trans. on Computer Systems*, Vol. 9, No. 4, pp. 364-373, November 1991.

[71] S. Keshav, "Packet-Pair Flow Control," *Technical Report, AT&T Bell Laboratories,* Murray Hill, NJ, 1994. Available via ftp://ftp.research.att.com/dist/qos/pp.ps.z.

[72] T. V. Lakshman and U. Madhow, "The Performance of TCP/IP for Networks with High Bandwidth-Delay Products and Random Loss," *IEEE/ACM Trans. Networking,* Vol. 5, No. 3, pp. 336-350, June 1997.

[73] J. R. Li, S. Ha, and V. Bharghavan, "HPF: A Transport Protocol For Supporting Heterogeneous Packet Flows in the Internet," in *Proc. IEEE INFOCOM 1999*, Vol. 3, pp. 543-550, March 1999.

[74] J. Liu, I. Matta, and M. Crovella, "End-to-end Inference of Loss Nature in a Hybrid Wired/Wireless Environment," in *Proc. Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt'03)*, Sophia-Antipolis, France, March 2003.

[75] D. Logothetis, K. S. Trivedi, and A. Puliafito, "Markov Regenerative Models," in *Proc. Int. Computer Performance and Dependability Symp.*, pp. 134-143, Erlangen, Germany 1995.

[76] S. Lu and V. Bharghavan, "Rate Adaptation in Cellular Packet Networks," *Internal Technical Report, TIMELY Research Group,* University of Illinois, http://timely.crhc.uiuc.edu, June 1997.

[77] S. Lu, K. -W. Lee, and V. Bhargavan, "Adaptive Service in Mobile Computing Environments," in *Proc. 5 th International Workshop on Quality of Service (IWQOS'97),* pp. 25-36, Columbia University, New York, USA, 1997.

[78] R. Ludwig and R. H. Katz, "The Eifel Algorithm: Making TCP Robust Against Spurious Retransmissions," *ACM Computer Communicaion Review*, Vol. 30, No. 1, pp.30-36, January 2000.

[79] M. W. Maeda, "Next Generation Internet: SuperNet Technology," available at *http://www.darpa.mil/ito/research/ngi/supernet.html*, July 2003.

[80] M. Mathis, J. Mahdavi, "Forward Acknowledgement: Refining TCP Congestion Control," *Proc. ACM SIGCOMM 1996* , pp. 281-292, Aug. 1996.

[81] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanov, "TCP Selective Acknowledgement Options," *RFC 2018*, 1996.

[82] MICA Motes and Sensors, *http://www.xbow.com/Products/ Wireless_Sensor_Networks.htm*

[83] J. Padhye, V. Firoio, D. Towsley, and J. Kurose, "Modeling TCP Reno Performance: A Simple Model and Its Empirical Validation," *IEEE/ACM Trans. Networking*, Vol. 8, No. 2, pp. 133-145, April 2000.

[84] C. Parsa and J. J. Garcia-Luna-Aceves, "Differentiating Congestion vs. Random Loss: A Method for Improving TCP Performance over Wireless Links," in *Proc. IEEE WCNC 2000*, Vol. 1, pp. 90-93, 2000.

[85] C. Partridge and T.J. Shepard, "TCP/IP Performance over Satellite Links," *IEEE Network*, pp. 44-49, September/October 1997.

[86] J. Postel, "DoD Standard Internet Protocol," *RFC 760*, January 1980.

[87] R. Puri, K-W. Lee, K. Ramchandran, and V. Bharghavan, "An Integrated Source Transcoding and Congestion Control Paradigm for Video Streaming in the Internet," *IEEE Trans. Multimedia*, Vol. 3, No. 1, pp. 18-32, March 2001.

[88] R. Rejaie, M. Handley, and D. Estrin, "RAP: An End-to-End Rate-based Congestion Control Mechanism for Realtime Streams in the Internet," in *Proc. IEEE INFOCOM 1999*, Vol. 3, pp. 1337-1345, March 1999.

[89] K. Salamatian, S. Vaton, "Hidden Markov Modeling for Network Communication Channels," in *Proc. ACM SIGMETRICS 2001*, pp. 92-101, Cambrigde, June 2001.

[90] O. Sallent, J. P-Romero, R. Agusti, F. Casadevall, "Provisioning Multimedia Wireless Networking for Better QoS: RRM Strategies for 3G W-CDMA," *IEEE Comm. Magazine*, Vol. 41, No. 2, pp. 100-106, February 2003.

[91] N. K. G. Samaraweera, "Non-congestion Packet Loss Detection for TCP Error Recovery Using Wireless Links," *IEE Proceedings-Communications*, Vol. 146, Issue 4, pp. 222-230, August 1999.

[92] Y. Sankarasubramaniam, O. B. Akan, and I. F. Akyildiz, "ESRT: Event-to-Sink Reliable Transport for Wireless Sensor Networks," in *Proc. ACM MOBIHOC 2003*, pp. 177-188, Annapolis, Maryland, USA, June 2003.

[93] R. Schnurr, J. Rash, K. Hogie, R. Parise, and E. Criscuolo, "NASA/GSFC Space Internet: Extending Internet Technology into Space," *NASA/GSFC Space Internet - NRO Technical Seminar*, NASA Goddard Space Flight Center, May 2001.

[94] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, "RTP: A Transport for Real-Time Applications," *RFC 1889*, January 1996.

[95] K. Scott and S. Burleigh, "Bundle Protocol Specification," *IETF Draft, draft-irtf-dtnrg-bundle-spec-01.txt,* October 2003.

[96] S. D. Servetto and G. Barrenechea, "Constrained Random Walks on Random Graphs: Routing Algorithms for Large Scale Wireless Sensor Networks," *In Proc. WSNA 2002,* pp. 12-21, Atlanta, GA, September 2002.

[97] S. Shenker, C. Partridge, and R. Guerin, "Specification of Guaranteed Quality of Service," *RFC 2212*, September 1997.

[98] E. Shih, S. Cho, N. Ickes, R. Min, A. Sinha, A. Wang, and A. Chandrakasan, "Physical Layer Driven Protocol and Algorithm Design for Energy-Efficient Wireless Sensor Networks," in *Proc. ACM MOBICOM 2001,* pp. 272-286, Rome, Italy, July 2001.

[99] P. Sinha, N. Venkataraman, R. Sivakumar, and V. Bharghavan, "WTCP: A Reliable Transport Protocol for Wireless Wide-Area Networks," in *Proc. ACM MOBICOM 1999*, pp. 231-241, Seattle, Washington, August 1999.

[100] D. Sisalem and H. Schulzrinne, "The Loss-Delay Based Adjustment Algorithm: A TCP-Friendly Adaptation Scheme," *Workshop on Network and Operating System Support for Digital Audio and Video*, July 1998.

[101] K. Sohrabi, B. Manriquez, and G. Pottie, "Near-Ground Wideband Channel Measurements," in *Proc. IEEE VTC 1999,* Vol. 1, pp. 571-574, New York, 1999.

[102] P. A. Spagnolo, T. R. Henderson, J. H. Kim, and G. T. Michael, "TCP Gateway Design Considerations for Satellite Link Blockage Mitigation," in *Proc. IEEE ICC 2003*, Vol. 1 , pp. 433-437, May 2003.

[103] F. Stann and J. Heidemann, "RMST: Reliable Data Transport in Sensor Networks," in *Proc. IEEE SNPA 2003*, pp. 102-112, Anchorage, Alaska, May 2003.

[104] W. Stevens, RFC 2001 - TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms.

[105] F. Swarts and H.C. Ferreira, "Markov Characterization of Digital Fading Mobile VHF Channels," *IEEE Trans. Veh. Technol.*, Vol. 43, No. 4, pp. 977-985, November 1994.

[106] J. Tang, G. Morabito, I. F. Akyildiz, and M. Johnson, "RCS: A Rate Control Scheme for Real-Time Traffic in Networks with High Bandwidth-Delay Products and High Bit Error Rates," in *Proc. IEEE INFOCOM 2001*, Vol. 1, pp. 114-122, April 2001.

[107] Test Model 5, TM5, *http://www.mpeg.org/MPEG/MSSG/tm5/.*

[108] The Delay-Tolerant Networking Research Group, *http://www.dtnrg.org.*

[109] The NASA Deep Space Network (DSN), *http://deepspace.jpl.nasa.gov/dsn/.*

[110] The National Space Science Data Center (NSSDC), NASA Goddard Space Flight Center, "Chronology of Lunar and Planetary Exploration," *http://nssdc.gsfc.nasa.gov/planetary/chrono.html.*

[111] The Network Simulator, ns-2, *http://www.isi.edu/nsnam/ns/index.html.*

[112] S. Tilak, N. B Abu-Ghazaleh, and W. Heinzelman, "Infrastructure Tradeoffs for Sensor Networks," in *Proc. ACM WSNA 2002,* pp. 49-58, September 2002, Atlanta, GA, USA.

[113] D. T. Tran, F. J. Lawas-Grodek, R. P. Dimond, W. D. Ivancic, "SCPS-TP, TCP and Rate-Based Protocol Evaluation for High-Delay, Error-Prone Links," *SpaceOps 2002*, Houston, TX, October 2002.

[114] E. Travis, "The Interplanetary Internet: Architecture and Key Technical Concepts," *Internet Global Summit,* INET 2001, June 5, 2001.

[115] M. Vojnovic and J. Y. Boudec, "On the Long-Run Behavior of Equation-Based Rate Control," *Technical Report IC/2002/06 EPFL*, Switzerland, June 2002.

[116] M. Vojnovic, J.-Y. Le Boudec, and C. Boutremans, "Global Fairness of Additive-increase and Multiplicative-decrease with Heterogeneous Round-Trip Times," in *Proc. IEEE INFOCOM 2000*, Vol. 3, pp. 1303-1312, April 2000.

[117] C. -Y. Wan, S. B. Eisenman, and A. T. Campbell, "CODA: Congestion Detection and Avoidance in Sensor Networks," in *Proc. ACM SENSYS 2003*, November 2003.

[118] C. Y. Wan, A. T. Campbell, and L. Krishnamurthy, "PSFQ: A Reliable Transport Protocol for Wireless Sensor Networks," in *Proc. ACM WSNA 2002,* pp. 1-11, September 2002, Atlanta, GA, USA.

[119] Y. Wang, J. Ostermann, and Y-Q. Zhang, "Video Processing and Communications," *Prentice Hall*, 2002.

[120] H. S. Wang and N. Moayeri, "Finite-State Markov Channel: A Useful Model for Radio Communications Channels," *IEEE Trans. Veh. Technol.*, Vol. 44, No. 1, pp. 163-171, February 1995.

[121] H. S. Wang, "On Verifying The First-Order Markovian Assumption for A Rayleigh Fading Channel Model," *IEEE Trans. Veh. Technol.*, Vol. 45, No. 2, pp. 353-357, May 1996.

[122] A. Woo and D. E. Culler, "A Transmission Control Scheme for Media Access in Sensor Networks," in *Proc. ACM MOBICOM 2001,* pp.221-235, Rome, Italy 2001.

[123] J. Wroclawski, "The Use of RSVP with IETF Integrated Services," *RFC 2210*, September 1997.

[124] D. Wu, Y.T. Hou, W. Zhu, Y.Q. Zhang, and J.M. Peha, "Streaming Video Over the Internet: Approaches and Directions," *IEEE Trans.Circuits Syst. Video Technol.*, Vol. 11, No. 3, pp. 282-300, March 2001.

[125] Y. R. Yang, M. S. Kim, and S. S. Lam, "Transient Behaviors of TCP-Friendly Congestion Control Protocols," in *Proc. IEEE INFOCOM 2001*, Vol. 3, pp. 1716-1725, March 2001.

[126] Y. R. Yang and S. S. Lam, "General AIMD Congestion Control," *Proc. IEEE ICNP 2000*, pp. 187-198, Osaka, Japan, November 2000.

[127] F. Yang, Q. Zhang, W. Zhu, and Y.-Q. Zhang, "An End-to-End TCP-Friendly Streaming Protocol for Multimedia over Wireless Internet," in *Proc. IEEE ICME 2003,* Vol. 2, pp. 429-432, 6-9 July 2003.

[128] W. Ye, J. Heidemann, and D. Estrin, "An Energy-Efficient MAC Protocol for Wireless Sensor Networks," in *Proc. INFOCOM'02,* pp. 1567-1576, New York, USA, June, 2002.

[129] M. Zorzi, R. R. Rao, and L. Milstein, "On The Accuracy of A First-Order Markov Model for Data Transmission on Fading Channels," in *Proc. IEEE ICUPC'95*, pp. 211-215, 1995.

[130] M. Zorzi, R. R. Rao, and L. B. Milstein, "ARQ Error Control for Fading Mobile Radio Channels," *IEEE Trans. Veh. Technol.*, Vol. 46, pp. 445-455, May 1997.

VITA

Özgür Barış Akan was born on November 28, 1977, in Ankara, Turkey. He received his B.Sc. and M.Sc. degrees in Electrical and Electronics Engineering from Bilkent University and Middle East Technical University, Ankara, Turkey, in 1999 and 2001, respectively. He attended the doctoral program at the School of Electrical and Computer Engineering of Georgia Institute of Technology, Atlanta, GA, from January 2002 to April 2004. At the same time, he was a research assistant in the Broadband and Wireless Networking Laboratory (BWN-LAB), Georgia Tech. He is also the recipient of the Researcher of Year 2003 Award of BWN-LAB. In April 2004, he received his Ph.D. in Electrical and Computer Engineering from Georgia Institute of Technology.